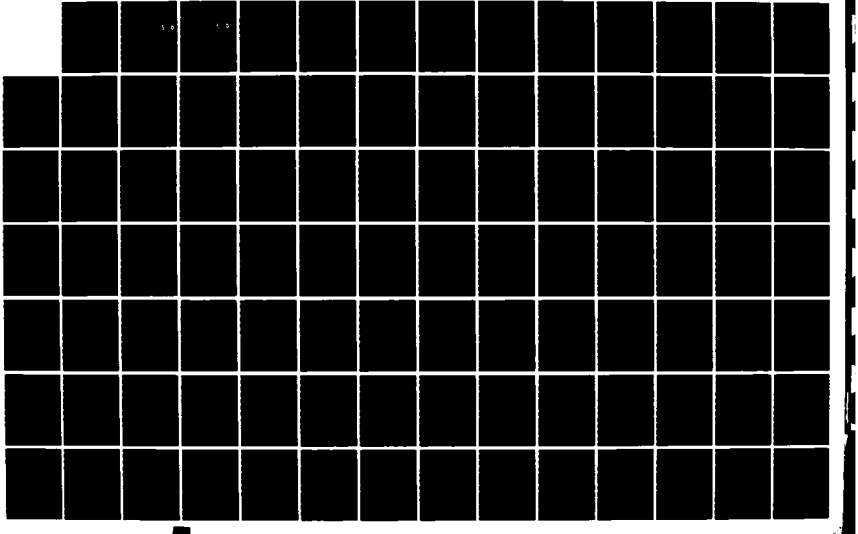
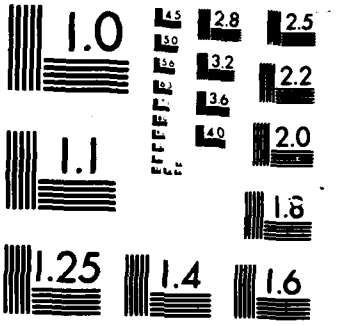
AD-A163 943 ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. P W COUTEE

1/2

UNCLASSIFIED DEC 85 AFIT/GE/ENG/85D-11 F/G 9/5 NL





ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR

THESIS

Paul W. Coutee
Captain, USAF

AFIT/GE/ENG/85D-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR

THESIS

Paul W. Coutee
Captain, USAF

AFIT/GE/ENG/85D-11

AFIT/GE/ENG/85D-11

ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Paul W. Coutee, B.S.
Captain, USAF

December 1985

Preface

Completion of this research effort would not have been possible without the full cooperation many individuals.

First, I thank my loving wife, Elizabeth, and our two sons, Eric and Adam. Their support and understanding made this achievement possible. I also express my gratitude for her administrative assistance in preparing this document.

A special recognition of appreciation to my faculty research advisor, Captain Richard W. Linderman, whose technical knowledge, assistance and team leadership kept all of the WFT research projects on course.

Thanks to Lt. Col. Harold W. Carter, for his encouragement and support in VLSI computer aided design tools. These tools are a must for any serious VLSI design project.

The enthusiasm and cooperation of the other WFT team members is greatly appreciated: Captains Jim Collins, Paul Rossbach and Kent Taylor.

Table of Contents

List of Figures

List of Tables

Abstract

This investigation defines, designs, and implements the arithmetic circuitry for a VLSI system capable of computing over eight thousand (8000) 4080-point Discrete Fourier Transform (DFT) problems per second with high numerical accuracy (over 100 db). An overview of the architecture illustrates how it is segregated into three smaller DFT problems, the 15, 16, and 17-point DFTs, using the Good-Thomas Prime Factor Algorithm (PFA). The smaller DFT problems are solved, using the Winograd Fourier Transform Algorithm (WFTA), on separate VLSI chips.

The chips were designed for a 1.2 μm CMOS process to operate at 70 MHZ. A detailed analysis of the serial pipeline architecture, used in the arithmetic sections to compute the WFTs, highlights the design of a constant coefficient serial pipeline multiplier. Each coefficient is encoded into its dual-bit (i.e. 0, ± 1, or ± 2 times 2^{2n}) equivalent and implemented using 6 multiplicand macrocells. The pipeline's pre- and post-addition stages feature a 24 device adder-subtractor cell designed to reduce the cell's load capacitance. The input and output circuitry perform: I/O data buffering, parity checking and generation, rounding, scale factor generation, and data word bit length extensions to prevent overflow and maintain numerical accuracy. Emphasis is placed on achieving a high measure of Fault Tolerance, Testability, and Low Power Consumption.

ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR

I. Introduction

1.0 Background

Digital Signal Processing (DSP) systems often employ the Discrete Fourier Transform (DFT) to convert between the space-time and frequency domains to do correlation and convolution problems [8:87-110; 9:356-419]. The number of arithmetic operations (i.e. additions, subtractions, and multiplications) required to directly compute the DFT is proportional to n^2, where n is the number of sampled data points. There are several algorithms which can be used to reduce the number of arithmetic operations to approximately n(log n) [1]. However, the time required to transform a large number of data points represents a significant congestion factor in many DSP applications. This is especially true in real-time operational systems, such as radar and image processing, where the number of sampled data points transformed must be large to achieve the required resolution.

A major research and design project was initiated to help reduce the DFT congestion factor in DSP systems. The goal is to implement a VLSI system which provides an

efficient means of computing the DFT for a large number of data points (i.e. up to 4080 points). The approach selected is centered around two algorithms, the Good-Thomas Prime Factor Algorithm (PFA) and the Winograd Fourier Transform Algorithm (WFTA). The PFA is used to segregate the 4080-point transform into three smaller sized transforms (i.e 15, 16, and 17 point transforms). Computation of the smaller transforms is accomplished via the WFTA which has the favorable characteristic of not only reducing the total number of operations, but also minimizing the required number of multiplications.

1.1 Problem Statement

The objective of this thesis is to define, design, and implement an efficient VLSI architecture which computes the Discrete Fourier Transform using the Winograd Fourier Transform Algorithm. The architecture includes circuitry to perform input/output, WFT calculations, parity checking and generation, and scale factor generation.

1.2 Scope

In addition to this thesis, the VLSI DFT research project encompasses three other research areas; (1) Algorithmic Theory/Numerical Analysis, (2) WFTA Control/ Address Generation Circuitry, and (3) Functional Simulation of WFT VLSI Circuits. The following sections provide a brief overview of each area.

1.2.1 Algorithmic Theory/Numerical Accuracy. This thesis topic is addressed by Captain Kent Taylor [12]. Captain Taylor's thesis presents a detailed description of the algorithmic theories employed in the design of the overall system level architecture, thereby illustrating the link between theory and hardware. This thesis also analyzes and discusses the numerical accuracy of the transformed data points.

1.2.2 WFTA Address Generation and Control Circuitry. This circuitry is analyzed in Captain Paul Rossbach's thesis [11]. The thesis provides a thorough analysis of the circuitry designed to generate the memory addresses for the input and output data points. Additionally, the circuitry employed to produce all of the control signals for the functional control of each WFTA chip is illustrated.

1.2.3 Simulation of VLSI Circuits. Captain Collins' thesis examines the utility and methodology of using the VHSIC hardware description language (VHDL) to simulate the functional operation of a WFT VLSI circuit [3]. This thesis also presents a detailed discussion of a custom program specifically designed to simulate the arithmetic circuitry of the 16-point WFTA. The program was written in the "c" programming language and provides an efficient means to validate the specified hardware design.

1.3 Overview of Thesis Contents

Chapter 2 contains a detailed problem analysis and presents some general information on the design approach that is applicable in Chapters 3 through 5. It begins with a discussion of the specific constraints imposed on the design and then proceeds to demonstrate how the constraints are incorporated into the design of the arithmetic circuitry. The material is presented in a manner which provides a top-down description of the overall system level design. It illustrates the efficiency obtained by employing a serial pipelined architecture to implement the arithmetic circuitry required to compute the WFTA. This chapter also contains descriptions of the basic microcells which are used to construct the macrocells contained within the pipeline.

Chapters 3 through 5 discuss the design and development of the major functional sections contained in the pipeline. To avoid repetitious discussion, similar functional sections are grouped and discussed within a chapter. The functional groupings and order of their presentations are as follows:

a. Chapter 3 - Multiplication Section

b. Chapter 4 - Pre-Addition/Post-Addition Sections

c. Chapter 5 - Input/Output Sections

Due to the large number of additions required to do multiplication, the efficient implementation of this section proved to be the key factor in obtaining the

desired functional throughput. One of the primary design obstacles was to define and then implement a serial pipelined multiplication architecture which did not consume an enormous amount of silicon area. Another obstacle constraint associated with the multiplication section is that it contains the worst case propagation delay in the pipeline.

Chapter 3 illustrates how these problems were resolved by taking advantage of the known constant coefficients associated with the WFTA and minimizing the capacitive loading. The discussion includes; (1) the encoding procedures used to form a dual-bit slice representation of each coefficient, (2) the arithmetic algorithms used to do signed two's complement multiplication with the multiplicand represented in a dual-bit slice code and the multiplier in binary form, (3) the hardware macrocells designed to perform the arithmetic algorithms, (4) the optimization schemes employed to reduce the silicon area and capacitive loads, and (5) the results obtained from SPICE transient analyses.

The primary macrocell employed in the pre- and post-addition sections is an adder-subtractor cell. Chapter 4 provides a detailed derivation of this cell whose most advantageous feature is the sharing of source and drain areas to reduce its total capacitive loading. It also demonstrates how these cells are integrated and controlled to perform matrix multiplication (where one matrix only

contains zeroes or ones) by chaining successive levels of additions and/or subtractions.

Chapter 5 discusses the Input and Output sections whose principle function is to provide a buffer for all data entering and leaving the pipelined chip. Additionally, these sections contain circuitry to; (1) perform parity checking and generation, (2) adjust the data word format on input and output data points, and (3) generate a scale factor code which identifies the largest magnitude associated with any one transformed data point in a block of 4080 data points. All of the macrocells cells required to accomplish these functions are examined from both a functional and hardware implementation viewpoint. This examination is presented in a manner which illustrates the interconnection and data flow between macrocells.

An overview of the results of this research effort is presented in Chapter 6.

II. Problem Analysis and WFTA Processor Overview

2.0 Overview

Designing a VLSI system, which computes large DFTs at a throughput rate sufficient for applications in radar and image processing systems, is complicated by the need to maintain a signal-to-noise ratio of approximately 120 db [12]. Further complications are introduced by imposing the additional requirement to incorporate the system characteristics of fault tolerance, low power consumption, and testability. This chapter addresses both the rationale for incorporating these additional features and the means employed to resolve all of the design complications. This information is presented in a manner which also provides a top-down overview of the entire system. The discussion leads to a description of the WFTA arithmetic circuitry architecture and its primitive microcells.

2.1 System Level Fault Tolerant Architecture.

It is desirable to incorporate fault tolerance in the VLSI DFT architecture to achieve high system reliability and maintainability. This enables the system to be operated in remote environments, such as space satellites, where maintenance accessability is minimal.

Fault tolerance is incorporated in the design at the system level through circuit redundancy and parity error monitoring. This technique is demonstrated in the following

paragraphs. First, a basic overview of the system level architecture is provided. Then, the system's circuit redundancy and parity checking features are discussed. This discussion will illustrate the system's flexibility to incorporate increasing levels of fault tolerance.

2.1.1 Basic 4080-Point DFT Architecture. The primary problem encountered in designing a VLSI system to directly and efficiently compute large discrete Fourier transforms (DFTs) is the enormous number of arithmetic operations required. This problem is resolved by implementing an integrated systems approach which uses two computationally efficient algorithms. First, the Good-Thomas Prime Factor Algorithm (PFA) is applied to convert a large one-dimensional DFT into a multi-dimensional DFT. Second, the Winograd Fourier Transform Algorithm (WFTA) is used to minimize the number of arithmetic operations required to compute each of the smaller DFTs. The Chinese Remainder Theorem provides the method of mapping the multi-dimensional result to a one-dimensional result used by the PFA.

The 4080-point DFT is converted into a multi-dimensional DFT requiring the computation of the 15, 16, and 17-point. Solving a 4080-point DFT in this manner requires the sequential calculation of 270 15-point DFTs, 240 17-point DFTs, and 255 16-point DFTs. The planned VLSI hardware system to accomplish these calculations is the

pipelined architecture shown in Figure 2.1. This system requires three WFTA chips which compute the 15, 17, and 16-point DFTs, and eight separate memory banks. Each memory bank has the capacity to store a complete 4080-point block transform (i.e. 4080 complex data points of 48 bits each = 163,200 bits).

Access control of each RAM is designed to enable each WFTA chip to continuously process data by alternating its input and output destinations after each block transform. For example, the 16-point WFTA chip alternates its input and output destinations between RAMs C & D and RAMs E & F, respectively. After each pass through a 4080-point block transform, the memory banks are swapped. A handshaking protocol between adjacent WFTA chips is employed to eliminate conflicts.

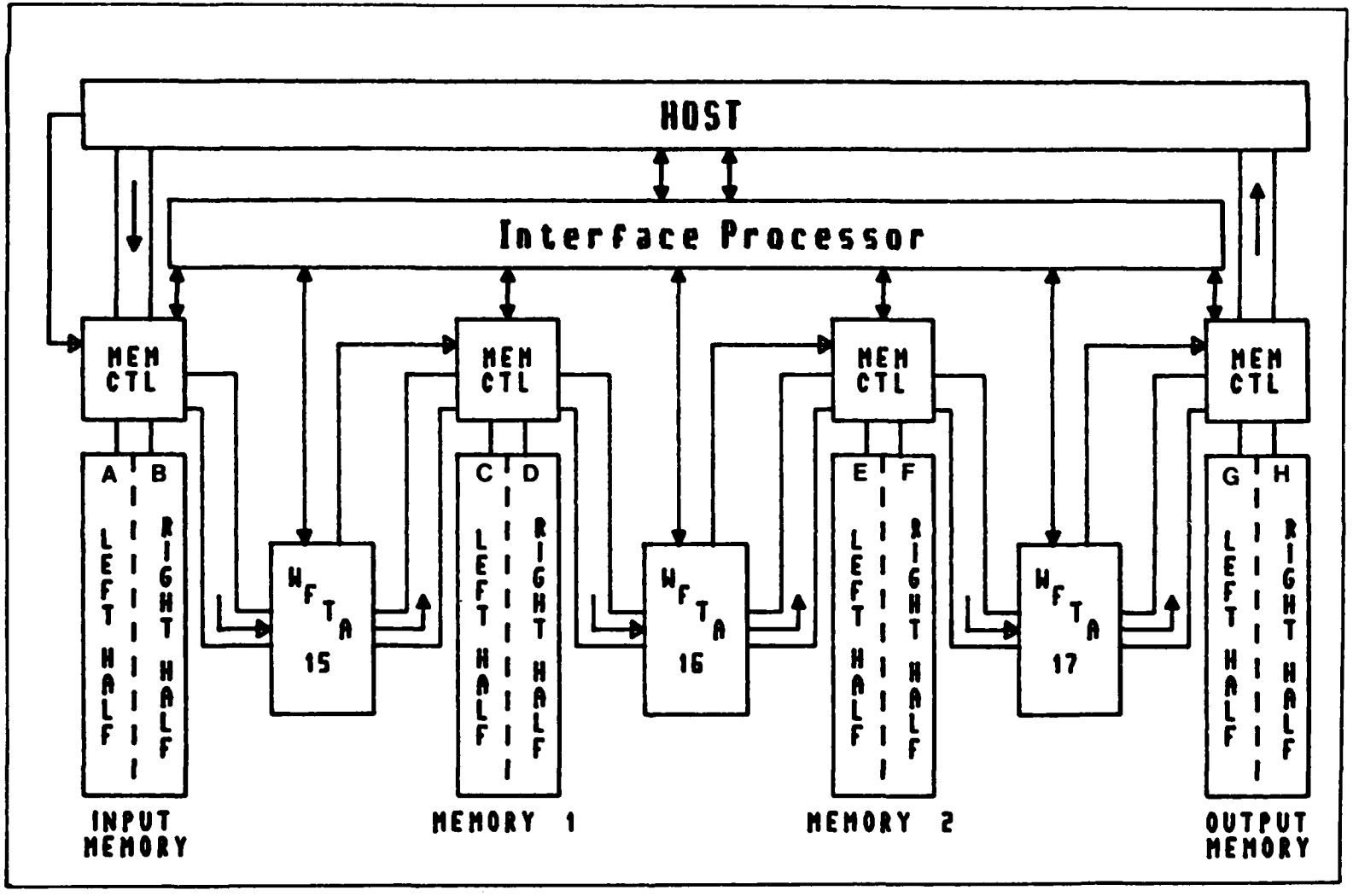


Figure 2.1. 4080-Point DFT Architecture

2.1.2 4080-Point DFT Architecture with Fault Tolerance. Fault tolerance is incorporated in the basic architecture by adding redundant WFTA chips and monitoring the chips' functional operation for parity errors. The resulting architecture is depicted in Figure 2.2. The additional chips are used to implement a watchdog scheme which is controlled by the host system. Under this scheme, all of the chips at each stage of the pipeline simultaneously receive the incoming data points and compute the Fourier transform. However, only one chip, designated by the host as the active chip, actually passes its transformed data points into the output memory bank. One of the remaining chips at each stage is designated the watchdog. As each transformed data point is output to the bus a comparison is made between the active and watchdog results. If the two values are not the same an error signal is generated and sent to the host system. The remaining chip at each stage can then be used to determine which chip is faulty by making further comparisons between it and the original active/watchdog chips. When the faulty chip is isolated, another chip is activated to take its place. Note that this scheme is not restricted to two redundant chips as shown in Figure 2.2. In general, the level of fault tolerance can be increased by incorporating additional redundant WFTA chips.

Monitoring the WFTA chips for parity errors provides a means to indirectly check the memory banks for faulty

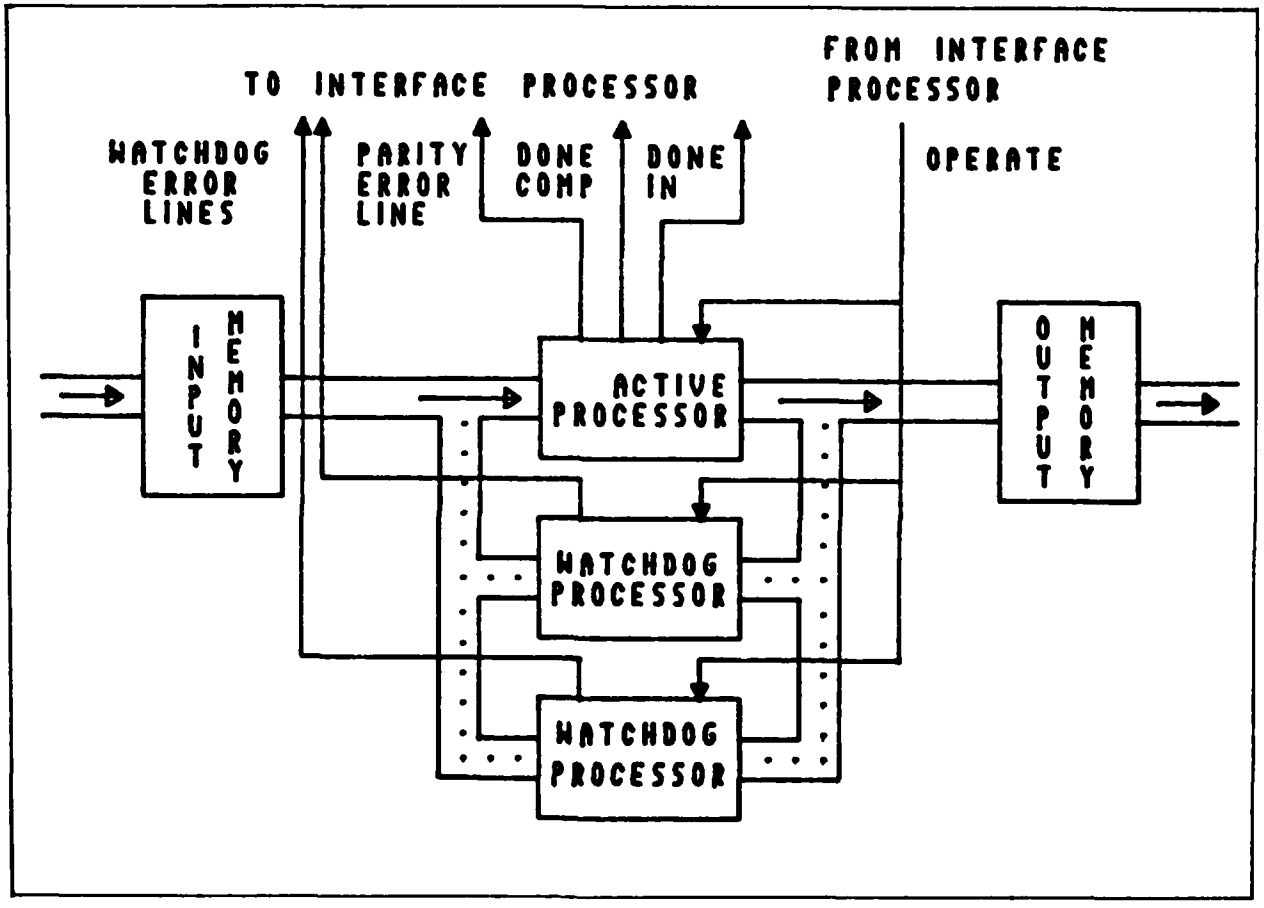


Figure 2.2. 4080-Point DFT Architecture with Fault Tolerance

storage locations. Accomplishment of this function requires a parity check on incoming data points and parity bit generation for transformed data points. A detailed description of the hardware circuitry used to perform parity checking/generation is presented in Chapter 5 of this thesis.

For more details on the algorithms, top level system design, and fault tolerance, the reader is referred to [11; 12].

2.2 High Functional Throughput Analysis

2.2.1 Problem Segregation at System Level. As previously stated, the reason for designing a VLSI DFT system with a high functional throughput is to eliminate the traditional digital signal processing bottleneck associated with computing large DFTs. Achieving a high functional throughput in any custom VLSI design requires careful consideration at each architectural level. The analysis and decision process begins at the system level for this design. As discussed in section 2.1, the first critical architectural decision involved segregating the 4080-Point DFT into three smaller problems, the 15, 16, and 17-point WFTs, which required fewer arithmetic operations. Also, the pipelined architecture enables the three smaller problems to be solved simultaneously, thus further increasing the functional throughput.

2.2.2 Efficient Architecture WFTA for Implementation. The next major architectural decision involved identifying the most efficient architecture to compute the WFTA. This decision was complicated by three constraints, numerical accuracy, speed, and silicon area. First, to achieve the desired numerical accuracy, which is driven by the targeted applications, the input and output data points must contain approximately 22-23 bits of precision (12). Therefore, the bit lengths manipulated by the WFTA arithmetic circuitry are fixed (see Chapters 3 and 4). The next two constraints, speed and silicon area, are normally dependent on each other. Designing circuits with exceptional speed normally requires more silicon area. This relates to the bounds placed on Area x Time complexity required for algorithms buried in VLSI [13]. A good example of this is parallel vs serial multiplication circuits where the parallel implementation is faster but requires significantly more silicon area to implement.

2.2.2.1 Parallel vs Serial Architecture. A comparison between the characteristics of the WFTA and the two fundamental architectures, parallel and serial, shows that the algorithm cannot be efficiently implemented using a parallel architecture. The principle difficulty encountered in trying to implement the algorithm using a parallel architecture is the excessive time required to store and access input data points, multiplicand coefficients, and intermediate results. Attempting to resolve this problem

by adding parallel multipliers and adder-subtractor circuits is not feasible because of the limited silicon area available. Storage and retrieval of the information on chip is constrained by time limitations. In either scenario, a complex data flow control problem must be resolved. Also, the chip becomes I/O bound and pin limited.

In contrast, a serial pipelined architecture provides an excellent correlation to the WFT algorithm. It provides a means to simultaneously transform all of the data points of a 15, 16, or 17-point problem. Also, DFT problems can be solved back to back within the pipeline. This makes efficient use of the hardware circuitry by keeping it 100 percent computationally active. Another efficient feature of the architecture is that very little control is required. Therefore, most of the silicon area can be used for calculation circuitry and less for control. All of these features, coupled with less I/O and an operational frequency of 70 MHZ, achieves the desired functional throughput [12].

2.3 Layout of WFTA Chip

The WFTA chip is partitioned into three functional sections, see Figure 2.3. The principle element in the input and output address generation circuitry is a read only memory (ROM) which is used to store the input and output data addresses. The function of the control

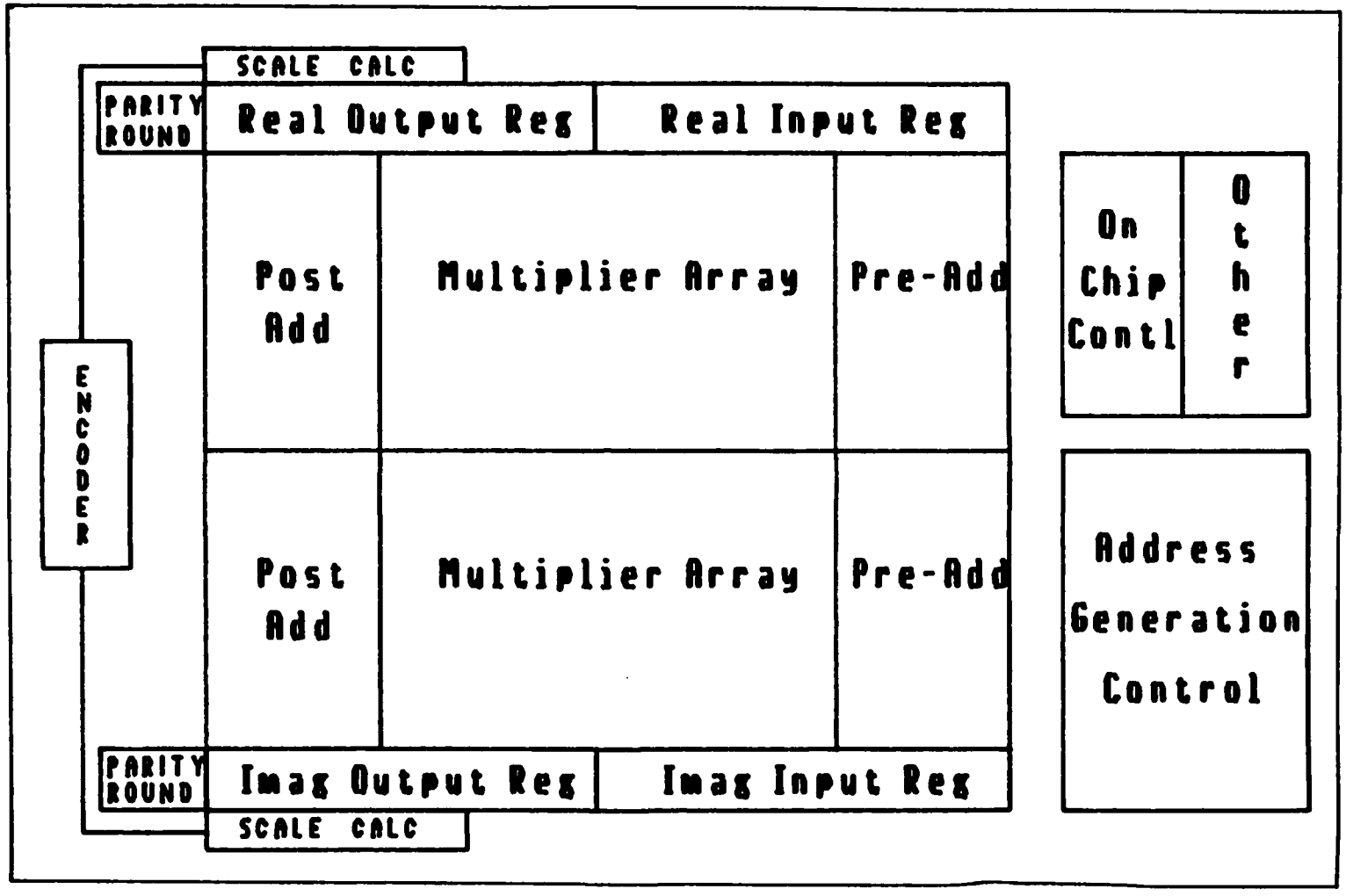


Figure 2.3. Layout of WFTA Chip

sequencer is to generate all of the required control signals to operate the chip. Its primary circuits are a ring counter and a programmable logic array (PLA) [11]. The last section is the WFT arithmetic circuitry. The remainder of this section provides a top level discussion of the arithmetic architecture. This discussion illustrates the architecture's efficient correlation to the WFT algorithm and briefly demonstrates the function of each major section.

2.3.1 WFTA Arithmetic Circuitry Architecture. The arithmetic circuitry architecture is a serial pipelined system which consists of three functional stages: input, output, and calculation. The complex data points input to the pipeline are segregated into two 24 bit words (i.e. one parity bit and 23 data bits) which define the real and imaginary magnitudes of each data point. Except for the post-addition stage of the arithmetic unit, the real and imaginary words are processed independently.

2.3.1.1 Input Stage. Beginning at the input stage, each data point is loaded, in a bit-parallel fashion, into an input buffer in blocks of 15, 16, or, 17 depending the DFT size being performed by the chip. When a complete block of data words have been loaded, they are latched into the serial pipeline and shifted through the arithmetic elements, least significant bit (LSB) first.

While still in the input unit, a parity check is performed to insure all inputs have odd parity, the input

parity bit is removed, and the bit length of each data word is extended to 30, 32, or 34 bits, depending on the size transform (i.e. 15, 16, or 17) implemented by the chip. One reason for extending the word length is to obtain an improved precision in the transformed results [12].

Another important reason for extending the word length is to prevent overflow conditions from occurring in the arithmetic unit. This extension is accomplished by adding zeroes at the least significant end of the word and sign extensions in its most significant positions. Chapter 5 of this thesis contains the specific details on the input unit.

2.3.1.2 Calculation Stage. The 32 bit data words out of the input unit are piped bit-serially into the arithmetic unit. This unit contains all of the arithmetic circuitry required to compute a DFT of a given size (i.e. either 15, 16, or 17-point). Using Winograd's Large and Small DFT Algorithms (i.e. large for 15-point DFT and small for 17 and 16-point DFTs), each of these calculations can be expressed in the following form:

$$V_o = A\ D\ C\ V_i \qquad (1)$$

where:

Vi is defined as the m x 1 matrix of input complex data points (m = 15, 17, or 16).

C is defined as the pre-addition matrix.

D is defined as the diagonal coefficient matrix.

A is defined as the post-addition matrix.
V_o is defined as the n x 1 matrix of transformed data points (complex).

(Note, the elements of matrices A, D, and C are different for each size transform).

Calculation of the transformed data points, V_o, is accomplished in three steps called pre-addition, multiplication, and post-addition. The first step, pre-addition, computes the product CV_i. All of the elements of the C matrix are +1, -1, or 0. So, the product CV_i is obtained by performing a series of additions or subtractions. For future reference, the product CV_i will be referred to as MR - the single column multiplier result matrix. In the second stage, multiplication, the matrix product DR is computed. As inferred from its name, the D matrix is a diagonal matrix whose diagonal elements are all real numbers that are scaled to be greater than or equal to negative one but less than positive one. Therefore, a multiplication algorithm must be employed to compute the product of D and MR. The result of this product is called the single column product result matrix and is labeled P_o.

The final step, post-addition, computes the product AP_o. Since elements of the post-addition matrix, A, are +1, -1, +j, -j, or 0, this product can also be obtained by performing a series of additions/subtractions. The results of this step are the transformed data points (i.e. V_o = a + jb). Each point is represented by two 32 bit words. One

word represents the real magnitude, a, and the other the imaginary magnitude, b. To summarize, the three steps used to perform the transform, $V_o = ADCV_i$, are:

a. pre-addition, $M_R = CV_i$

b. multiplication. $P_o = DR$

c. post-addition, $V_o = AP_o$

Further details on the derivation of the A, D, and C matrices is available in [12].

2.3.1.3 Output Stage. In the first phase of the output stage, the 30, 32, or 34-bit data words received from the arithmetic unit are changed into 24-bit words. This process is accomplished by implementing a rounding operation at bit position 7, 9, or 11 of the 30, 32, or 34 bit data words, respectively. After rounding, the least significant bits are dropped. Based on the remaining 23 most significant bits (MSB), a parity bit is generated and appended at bit position 24 to form the 24-bit output data words. These words are loaded bit-serially into buffer registers. Then they are latched into an interconnected stack of parallel registers and output to the data out pads two words at a time, one real and one imaginary. Also, as the words are transferred to the data out pads, a scale factor associated with all of the 4080 transformed data words is computed. This scale factor identifies the minimum number of sign bits contained in any of the 4080 words.

2.4 Testability of WFT Arithmetic Circuitry

An important consideration in any VLSI design is testability. As the complexity and number of devices in VLSI designs increase, the functional chip yield per wafer decreases. Also, chip packaging is a major cost factor. Therefore, the ability to detect and eliminate inoperable VLSI circuits before packaging can allow significant cost savings.

A primary goal is to include testability in the design in a manner which limits the additional hardware required and does not degrade circuit performance. Test structures must provide a high degree of controllability and observability during testing. Test circuits on the WFT processor accomplish this goal by taking advantage of its serial pipelined architecture. The testing method employed is similar the Level Sensitive Scan Design technique which was first introduced by the IBM corporation [2]. The 16-point WFT processor will be used as an example to demonstrate the planned testing method.

The preliminary test structure for the 16-point WFT is depicted in Figure 2.4. When the test signal (T) is high, multiplexers at the input and output pads are used to direct data to and from the two test buses. The T-gate configurations between each functional block enable test vectors to be input to any functional block and the resulting outputs observed.

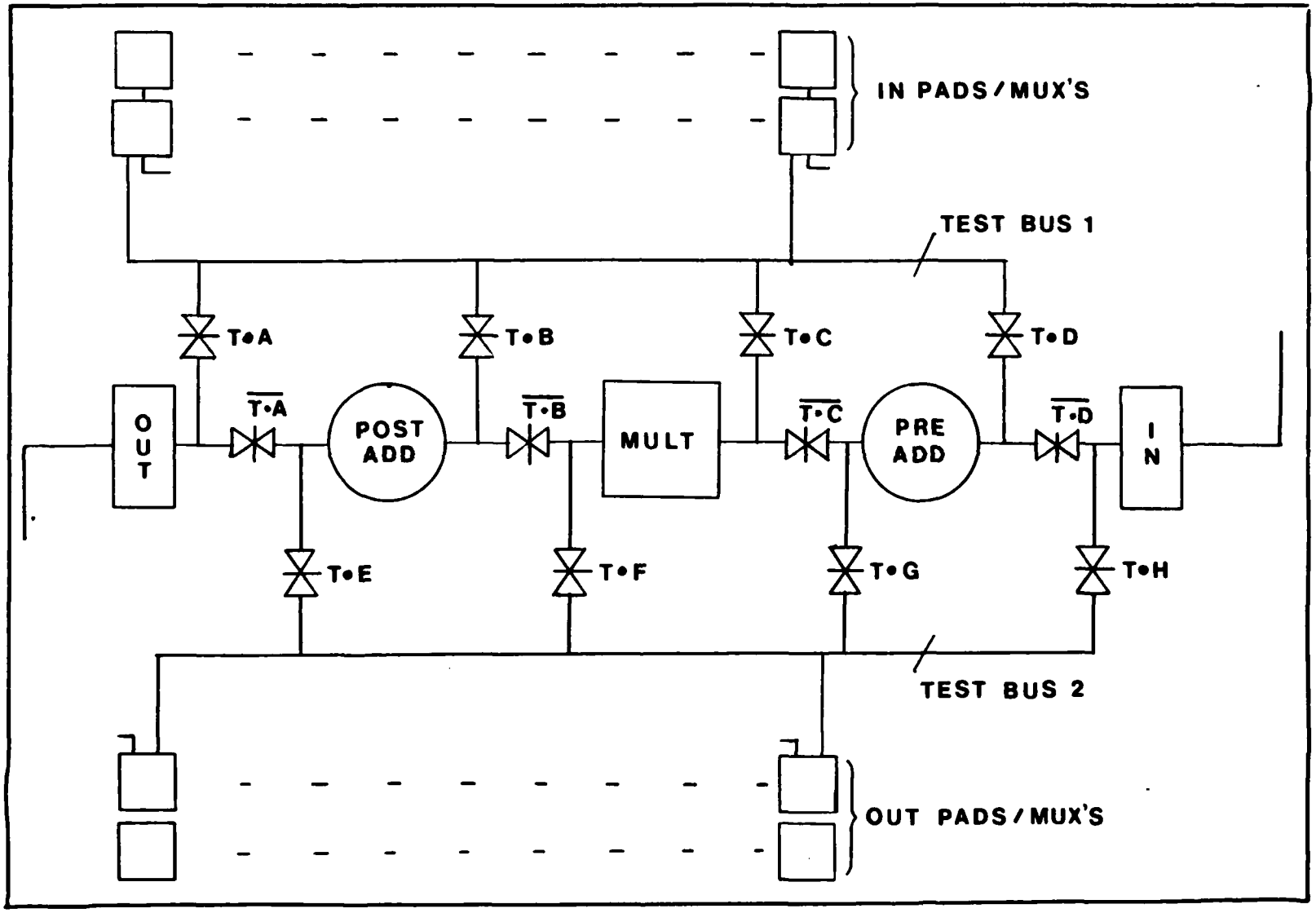


Figure 2.4. Arithmetic Circuitry with Testability Hardware

2.5 Selection of CMOS Technology

Currently, two technologies dominate VLSI circuit design. They are n-channel metal oxide semiconductor (NMOS) and complementary metal oxide semiconductor (CMOS). The technology selected for the design of the WFTA chips was CMOS. This choice was largely due to the difference in power dissipation between the two technologies. During logic transitions, power is dissipated in both. However, under static conditions almost no power is dissipated in CMOS gates. In contrast, anytime the output of an NMOS gate is logical "0", static power is consumed. The resultant heat generated in NMOS integrated circuits is hard to remove and can degrade the performance of the transistors. As a consequence of the higher power dissipation, designing VLSI NMOS circuits requires complex circuit forms to reduce the power dissipation [4:253].

Another potential problem in VLSI designs that is directly related to excess static power dissipation is metal migration. This is a physical process that can cause open and short circuits in metal layers. Several studies have concluded that metal migration is worse in circuits subjected to steady DC currents (as opposed to pulsed currents) and high temperatures [4:134; 8:53]. Therefore, the lower static power advantage of CMOS circuits reduces the probability of metal migration.

A further advantage of CMOS circuits is that their output logic levels are fully restored (i.e. the output

settles at V_{DD} or GND). For classical CMOS designs, where there exists one p-channel device for each n-channel device, the fully restored logic levels lead to excellent noise immunity [4:211].

The primary disadvantage of CMOS logic design is that it often requires more devices to implement a given function. To implement an N input gate, complementary static CMOS designs require 2N devices and NMOS designs require N + 1 devices [14:28].

Another hindrance associated with CMOS is latch-up. When intermixing both p-channel and n-channel devices on the same silicon substrate, it is possible to incorporate and inadvertently activate thyristor devices in a manner which shorts the supply voltage to ground. This condition is called latch-up and can cause functional failures and possibly permanent damage to the chip. In the early years of CMOS development, latch-up was a frequent occurrence and a serious problem. Fortunately, improved processing techniques, more stringent design rules, and designer awareness have combined to enable the designer to resolve potential latch-up conditions. For specific details on latch-up and techniques employed to avoid it, the reader is referred to [14:58; 4:123].

2.6 Device Sizing Considerations

Another means to reduce power consumption is to use

minimal device sizes where possible. However, the relative ratio between p and n type devices directly effects other circuit performance parameters. Specifically, appropriate device sizing is a key consideration in estimating CMOS circuit performance in the areas of noise immunity and switching delays. Therefore, determining a minimum size ratio must take these areas into consideration. It is standard practice to use the CMOS inverter as a baseline for demonstrating the relationship between device sizing and the above circuit characteristics. The following paragraphs provide a brief overview on the results of such analyses. For specific information on the derivation of these results, the reader is referred to [4; 5; 14].

The switching point of a CMOS inverter can be defined as the point at which both devices are operating in their saturation regions. Under these conditions and the assumption that the drain to source currents, I_{ds}, are essentially constant, it can be shown [14:47] that the input voltage is given by:

$$V_{in} = \frac{V_{DD} + Vtp + V_{tn}\sqrt{\frac{\beta_n}{\beta_p}}}{1 + \sqrt{\frac{\beta_n}{\beta_p}}} \qquad (2)$$

where: V_{tp} = threshold voltage of p-device
v_{tn} = threshold voltage of n-device
β_p = p-device gain ratio
β_n = n-device gain ratio

and the output voltage is in the range:

$$V_{in} - V_{tn} < V_o < V_{in} - V_{tp}$$

setting $\beta_n = \beta_p$ and $V_{tn} = V_{tp}$ gives:

$$V_{in} = \frac{V_{DD}}{2} \qquad (3)$$

As illustrated in Figure 2.5, these results imply that when both devices are in saturation, there is an infinite slope in the V_o vs V_{in} curve. However, there is a slight increase in I_{ds} in the saturation region. Therefore, a small negative slope is present.

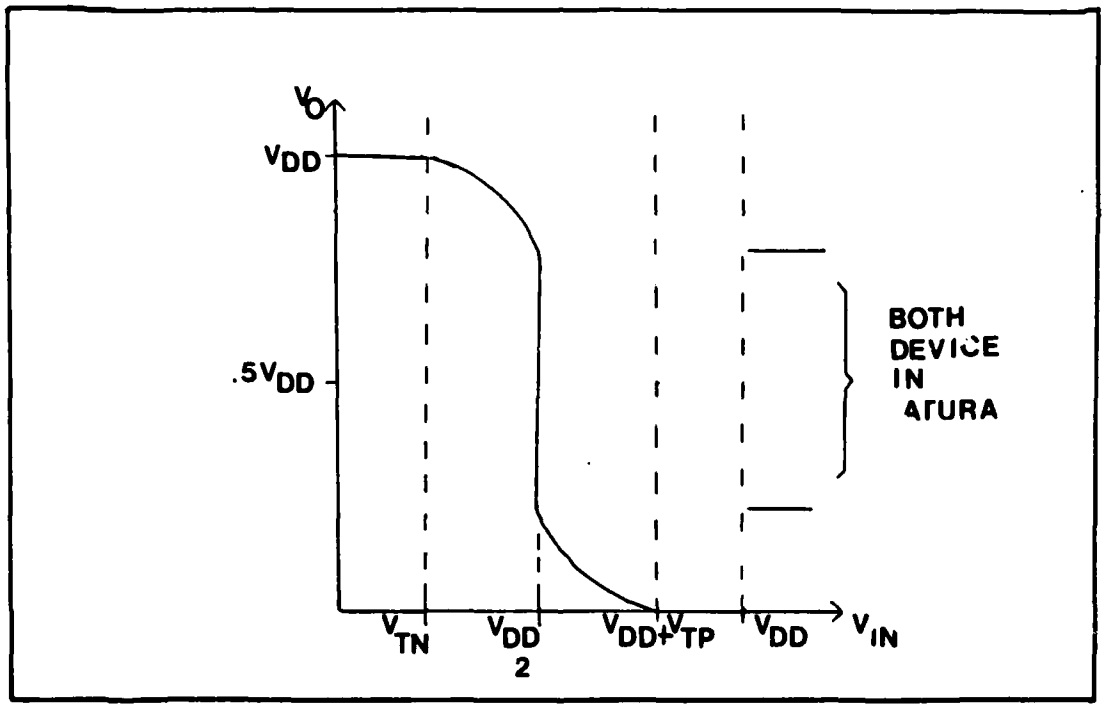


Figure 2.5 CMOS Switching Characteristics

The important point to note from the above results is the symmetry about the switching point, $V_{dd}/2$, and the sharp transition between logic levels. This gives rise to

efficient and symmetrical switching characteristics. As derived in the literature [14:137], the rise and fall times of a CMOS inverter (with $V_{tn} = V_{tp} = 0.2V_{DD}$ and V_{DD} = 5 volts) is given by:

$$t_r = \frac{4\ C_L}{\beta_p\ V_{DD}} \tag{4}$$

$$t_f = \frac{4\ C_L}{\beta_n\ V_{DD}} \tag{5}$$

where: t_r is the time required for the output to rise from 10 to 90 percent of its final value.

t_f is the time required for the output to fall from 90 to 10 percent of its final values

C_L is the load capacitance at the output

Note that if the devices' respective gains are equal ($\beta_n = \beta_p$) then the rise and fall times will be equal.

Given the same conditions (i.e. equal gains and equal threshold voltages), it can be shown that both the low and high noise immunities (NM_L and NM_H) are approximately equal to 0.425 x V_{DD}.

In each of the above cases, the necessary condition for symmetry was equal gains and equal thresholds. Since the threshold values are a direct function of the fabrication process, there is little the designer can do to effect their change. The device gains are not only a function of the fabrication process, but also the design layout. They are given by the following equations:

$$\beta_n = \frac{\mu_n \ (W_n/L_n)}{t_{ox}} \qquad (6)$$

$$\beta_p = \frac{\mu_p \ (W_p/L_p)}{t_{ox}} \qquad (7)$$

where: μ_n = surface mobility of electrons in the channel

μ_p = surface mobility of hole in the channel

t_{ox} = thickness of gate insulator

= permittivity of gate insulator

W_n = gate width of the n-device

W_p = gate width of the p-device

L_n = gate length of the n-device

L_p = gate length of the p-device

Using a 4 lambda width diffusion for n-devices and the approximate mobility values attained from MOSIS process parameters, equations 6 and 7 are set equal to each other. These equations are then reduced to show that the n to p width ratio is approximately 4/7. The WFTA chip employs this relative width ratio throughout the design.

2.7 CMOS Microcells Used in WFTA Arithmetic Circuits

This section presents a device level discussion on the microcells used in the WFTA processor. These cells are divided into two general categories, basic logic gates and storage registers. Their gate (or block) level representa-

tions are used in Chapters 3 through 5 to construct and explain all of the macrocells in the WFTA processor.

2.7.1 Static CMOS Logic Gates. As with most designs, the WFTA processor employs an integrated system of macrocells to perform the WFTA. These macrocells are constructed by interconnecting microcells which perform basic logic functions. The function and circuit description of the microcells are summarized in Figure 2.6. Each cell is depicted in three configurations. These representations provide each cell's description at the gate level, standard device level, and a general layout configuration. Some cells shown are implemented in both standard complementary CMOS logic and transmission gate (T-gate) logic.

The potential differences between the two logic implementations of a given function include:

a. The number of devices required to implement the function.

b. The sense (i.e. true or complement) of the input signals required to produce the function.

c. The relative switching delay.

d. The general layout configuration.

The particular design used at any given point is a function of the interfacing circuitry. For example, if an OR gate is needed, the first choice might be its T-gate version because it only requires three devices. But, if the true sense of the A input were not immediately available, it would have to be either formed with an inverter (adding

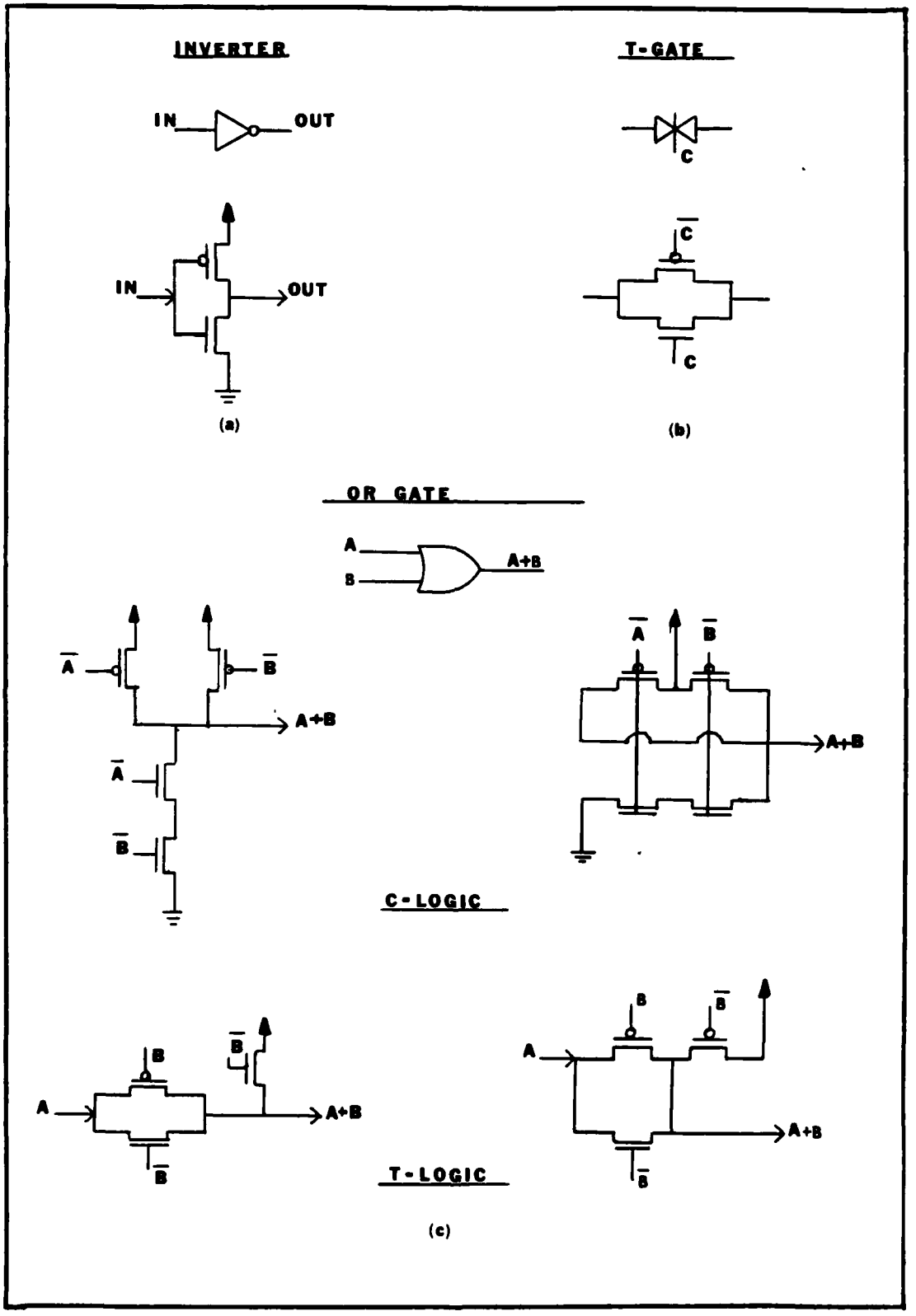


Figure 2.6. Static CMOS Logic Gates

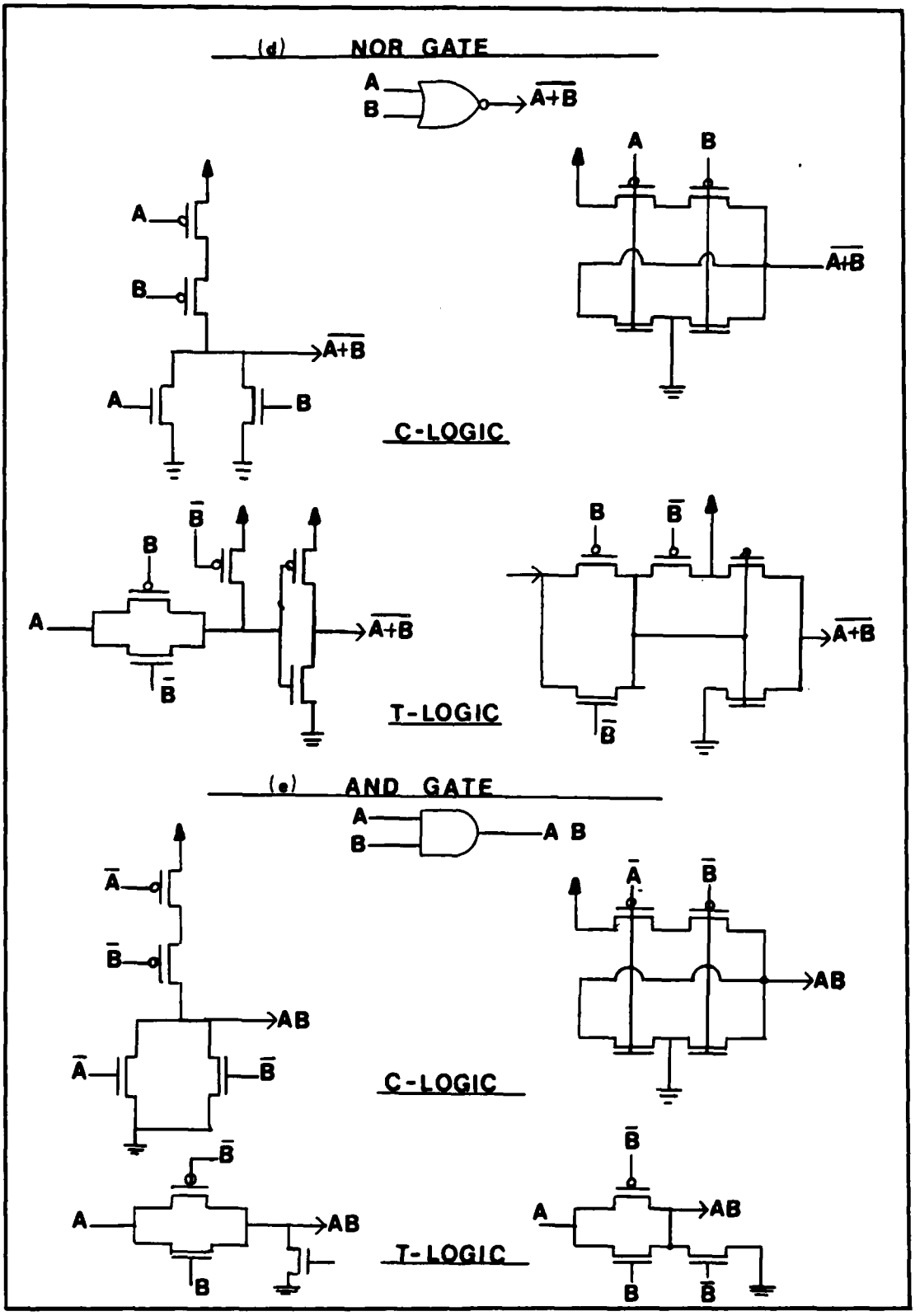


Figure 2.6 (continued)

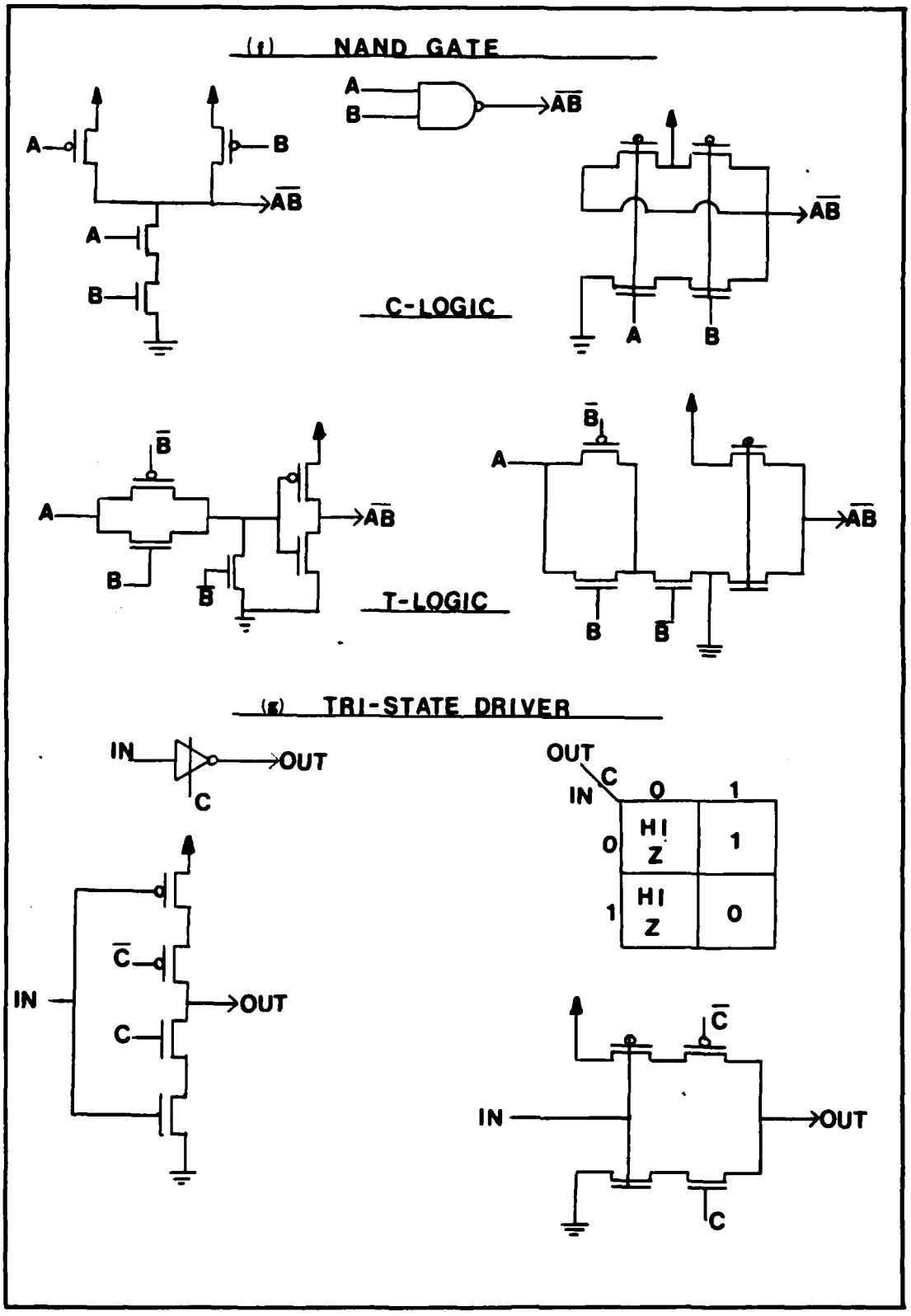


Figure 2.6 (continued)

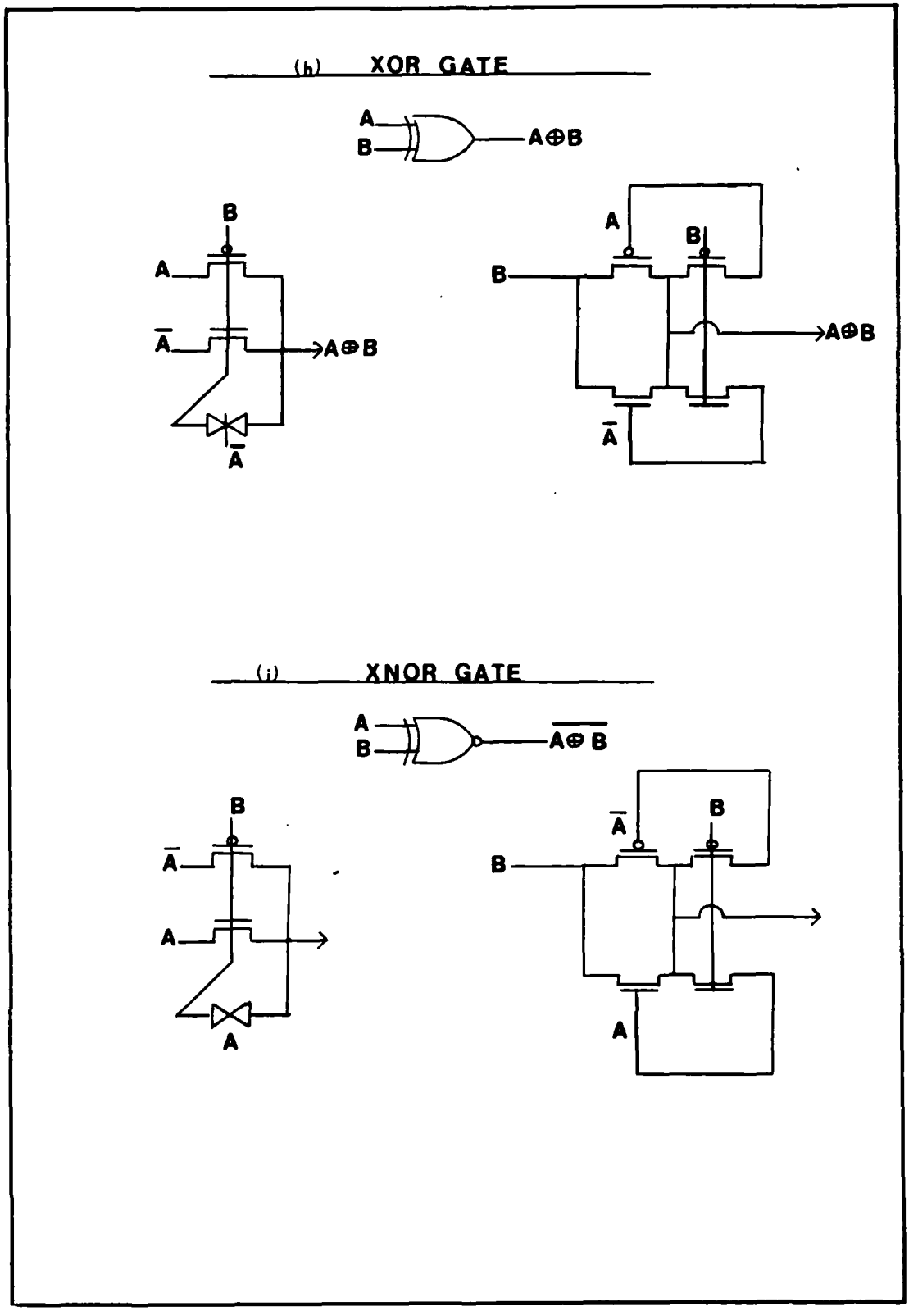


Figure 2.6 (continued)

two more devices) or perhaps routed in from a long distance. Under these conditions the complementary CMOS version might be used. Many similar scenarios are encountered in the design process and must be resolved by the designer.

There are two undesirable T-gate configurations. First, as a result of channel pinch-off, the n-channel device passes a degraded logical "1", typically 3.8 to 4 volts, and the p-channel device passes a degraded logical "0", approximately 1 to 1.2 volts. An example of this is illustrated in Figure 2.7(a) where an XOR gate is constructed using only two devices. Another common error in T-gate design is imposing a condition called fighting. This occurs when the outputs of two or more T-gates attempt to drive the same node to different logic levels. An example is shown in Figure 2.7(b).

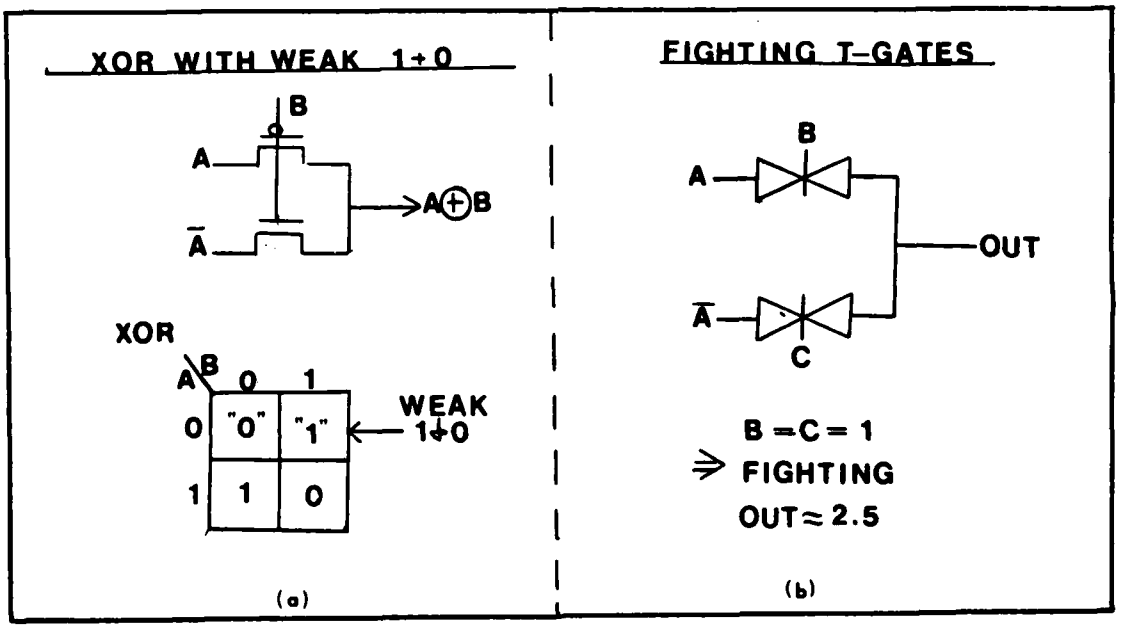


Figure 2.7 Abnormal T-Gate Configurations

2.6.2 Storage Registers. The WFTA processor employs a dual-phase (phi1 and phi2) non-overlapping clocking scheme to control the propagation of data through the pipeline. The data paths are constructed using master-slave flip-flops (MSFF) and phi1/phi2 latches. These registers control the inputs/outputs into the combinational logic circuits, provide delay elements, and store the current states of finite state machines. Using MSFFs controlled by a two-phase, non-overlapping clock as opposed to standard D type flip-flops controlled by a single phase clock eliminates the potential for race conditions to develop in the combinational logic. The two-phase clock also provides an extra dimension of control because there are two separate inputs to the chip. The symbolic representations of registers used in Chapters 3 through 5 are provided in Figure 2.8.

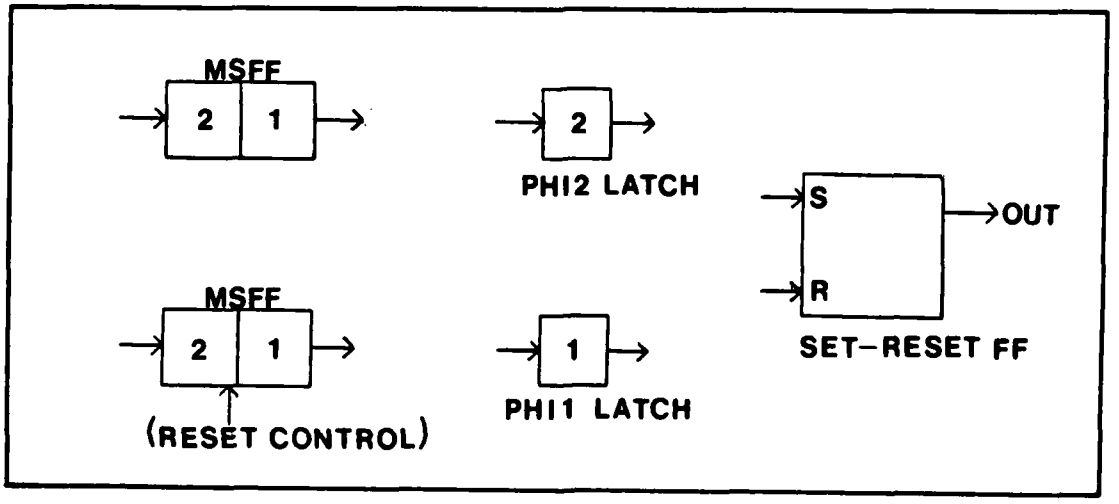


Figure 2.8 Block Symbols for Storage Registers

The gate level configurations of the various forms of storage registers employed in the WFTA processor are shown in Figure 2.9. The static MSFFs are used in locations where the data is not refreshed every clock cycle. Examples of their use would be the parallel paths of the input and output buffer cells (SIPO/PISO), discussed in Chapter 3. Dynamic MSFFs are predominately used in serial data paths where the data stored is refreshed every clock cycle. The use of dynamic MSFFs wherever possible has two advantages. It reduces the number of devices required to implement a function and reduces the capacitive load on the clock lines. Resettable MSFFs are used to initialize the configuration of functional cells throughout the WFTA processor.

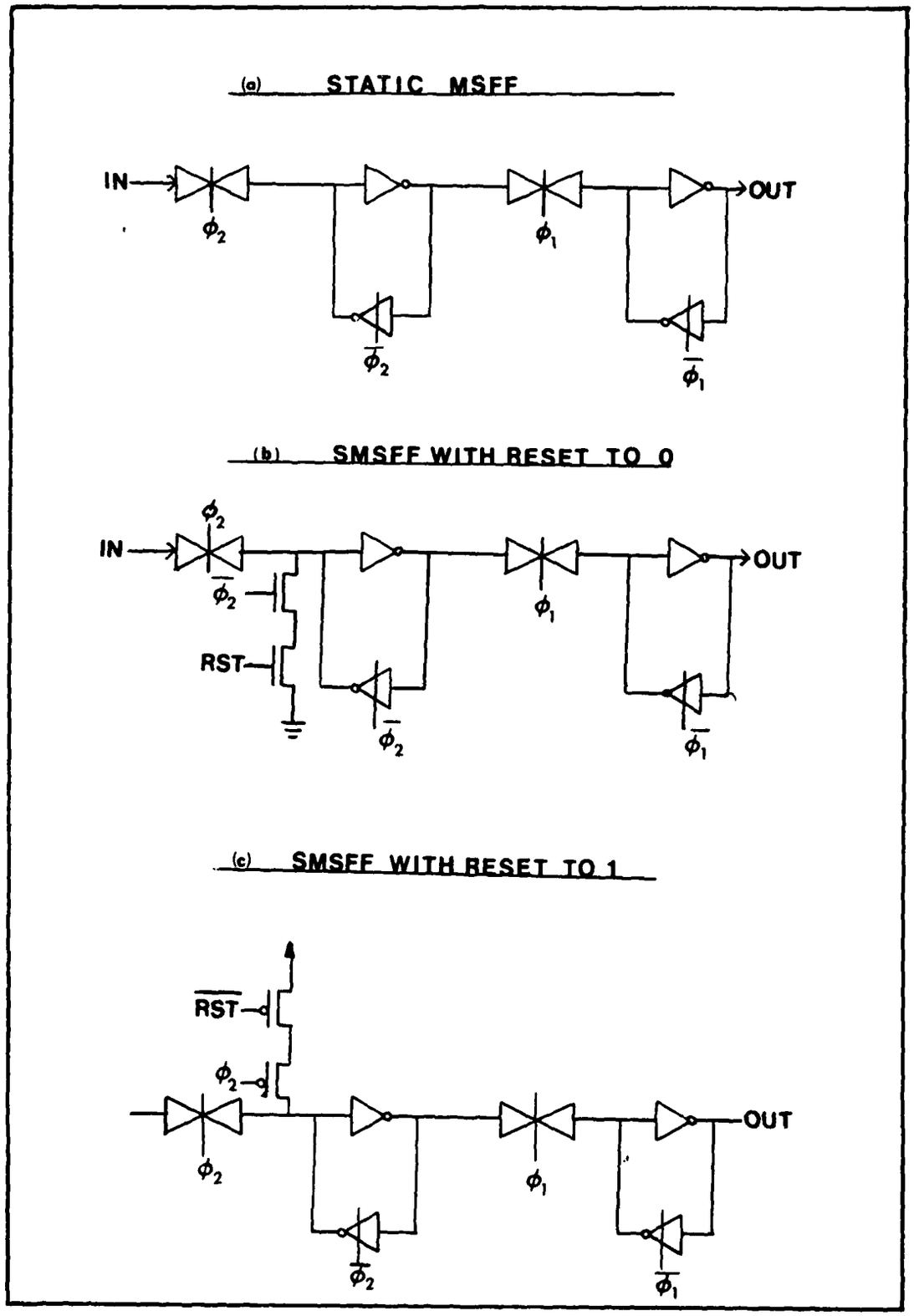


Figure 2.9. Gate Level Configuration of Storage Registers

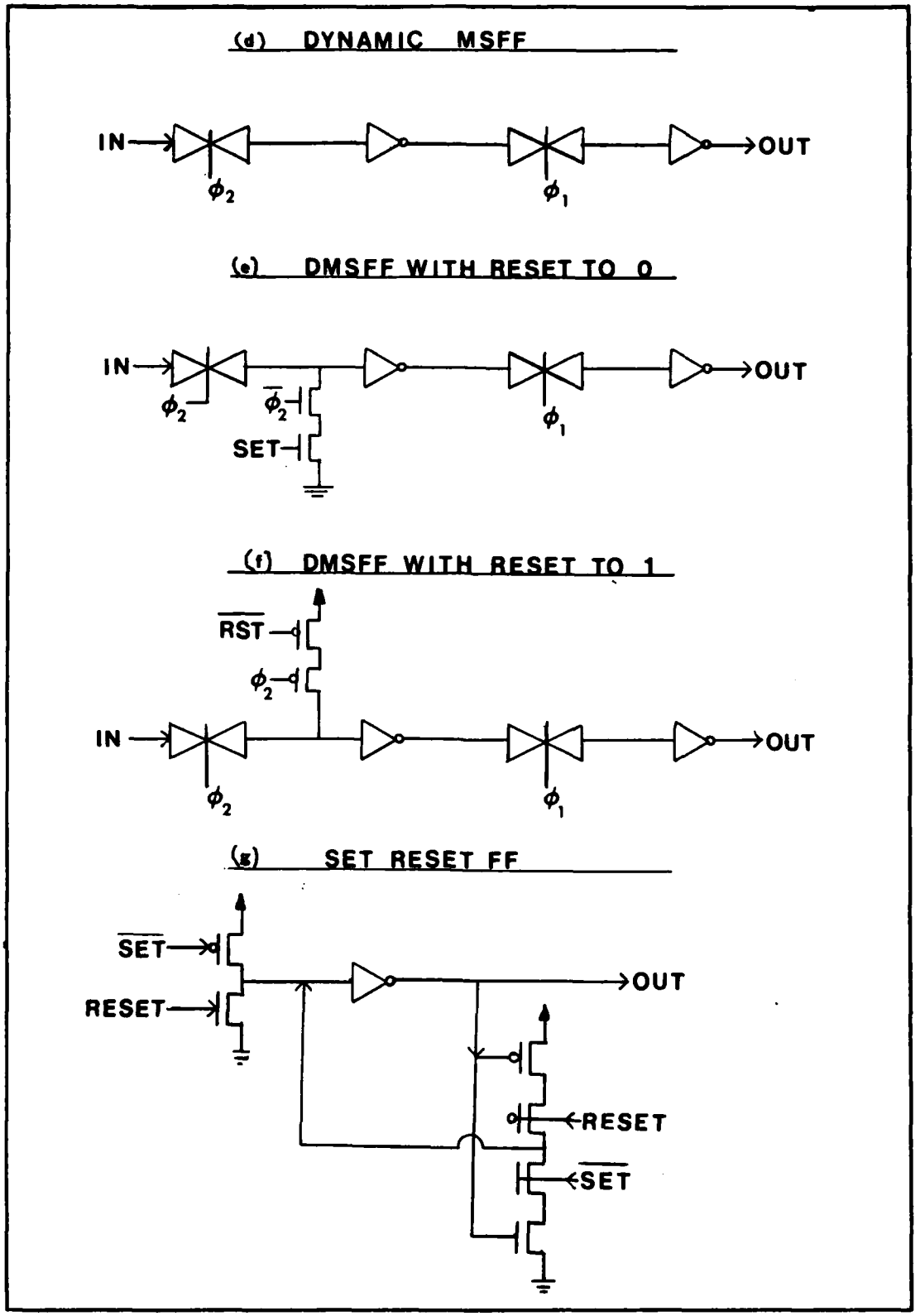


Figure 2.9 (continued)

III. WFT Multiplier Stage

3.0 Overview

The objective of the multiplier stage is to compute the single column multiplication result matrix, P_O = [D] X [MR] at an operational speed of 50 to 70 MHZ. To achieve this objective, the computation is carried out in signed 2's complement arithmetic via a pipelined multiplier architecture. To illustrate this architecture the 16-point WFTA implementation is discussed. However, the same general layout and macrocells are applicable to the 15 and 17-point architectures. The only difference between the three are the size of the cell arrays and the adjusted bit lengths of the data words.

For the 16-point transform, the MR matrix is 18 x 1 and the diagonal coefficient matrix , D, is 18 x 18. Therefore, simultaneous computation of all 16 complex data points requires 36 parallel multiplications (i.e. 18 for both the real and imaginary parts of the data word). The multipliers (i.e. the MR matrix elements) in this process are the 32 bit-serial data streams received from the pre-addition stage. The multiplicands are the fixed coefficients of the diagonal matrix.

By using Booth's Quaternary encoding algorithm, these coefficients are transformed into 14 dual-bit hardwired multiplicand modules which perform successive partial product operations on the input multiplier data streams.

There are five distinct dual-bit multiplicand modules whose values are $+2 \times 2^{2n}$, -2×2^{2n}, $+1 \times 2^{2n}$, -1×2^{2n}, and zero. The value of n is determined by the relative position of the module in the pipeline. Each coefficient is determined to an accuracy of 28 binary bits and is then recoded to its dual-bit slice code. Therefore, the range of n is 0,1,2,3, 4,.....,13. After application of Booth's algorithm, another encoding modification may be required to insure overflow errors do not occur during the multiplication process. Also, inherent in the module designs is the ability to perform rounding. This feature is important because the 32-bit blocked pipeline architecture will only provide a 32-bit result. Therefore, if the result is rounded as opposed to truncated, an extra degree of accuracy is achieved.

Operational control of the multiplication process is provided by a module called MULTCONT. This module generates all of the control signals required for the proper operation of the five multiplicand modules. Only one control module is required for each column of the multiplicand array. The control signals are activated by a single pulse generated by the control sequencer.

The remainder of this chapter provides a detailed analysis on the development of multiplication pipeline, a presentation of the 16-point multiplication pipeline, and a discussion on an additional encoding technique investigated in an attempt to maximize the number of zero modules in the

pipeline.

3.1 Multiplier Encoding

Four steps are required to obtain the final encoded representation of each diagonal element in the coefficient matrix, D. The first is to calculate a scaled decimal equivalent for each coefficient, d_x, in the range $-1 < d_x < +1$ (where $x = 1,2,3,...,m$ and m is a function of the size of the transform being computed). Unscaled decimal coefficients are computed using the WFT algorithm. Then, based on the largest absolute value obtained for any d_x, all of the coefficients are scaled down by 2^n (where n is large enough to insure that the coefficients are in the range specified above). For more information on calculating the initial decimal coefficients, the reader is referred to [12].

The second encoding step is to convert the scaled decimal coefficients into their signed 2's complement binary equivalents. To obtain more accuracy, the initial conversion is to a bit length of 30-bits, and is then rounded to 28-bits.

In the third step of the encoding process, Booth's Quaternary algorithm (also called dual-bit and bit-pair encoding) is applied to the above results. The advantage of doing this is two fold. First, it transforms both positive and negative multipliers into forms that enable a direct multiplication process which produces 2's complement

results. Second, it guarantees that there will be a maximum of n/2 partial product calculations required for an n-bit multiplicand [15:148-163]. Therefore, the number of modules required for the pipeline is cut in half. The rules for encoding the 2's complement bit stream are summarized in Table 3.1. As shown, the resulting dual-bit data stream can contain five different values (+1, -1, +2, -2, and 0). Derivation of these rules can be found in [15:148-163].

a_{n+1}	a_n	a_{n-1}	Dual-Bit Code
0	0	0	0
0	0	0	+1
0	1	0	+1
0	1	1	+2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

where a_n is the bit at location 2_n and n = 0,2,4,6,8,....

Table 3.1 Booth's Quaternary Algorithm Rules

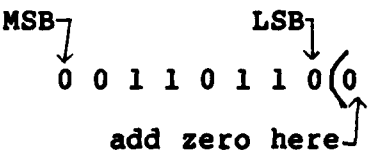
A simple procedure to encode a given data bit stream is as follows:

a. Re-Format Bit Stream: Beginning at the least significant bit (LSB), partition the bits in pairs of two as shown in the example below. If there is an odd number of bits, do a sign extension of the most significant bit (MSB) to form the last pair. Place a zero to the right of the LSB.

b. Determine Three-Bit Transitional Value: Consider the bits in groups of three, where each group is formed by the partitioned pairs and the first bit to their right. Starting with the LSBs (assume a "0" to the right of LSB) and moving from right to left, identify the transitions from 0 to 1 and 1 to 0. Either transition is interpreted to represent a magnitude of one when moving from bit 1 to 2 and a magnitude of two when moving from bit 2 to 3. The magnitude is zero when no transition occurs between bits. The sign associated with each magnitude is negative for 0 to 1 transitions and positive for 1 to 0 transitions.

c. Determine Dual-Bit Code: The elements of the multiplicand dual-bit code are given by the algebraic sum of each three-bit transitional value obtained in step b.

Example: Find the dual-bit code of the 2's complement data stream 0110110 (= +54) using the procedure outlined above.



	3-2-1	3-2-1	3-2-1	3-2-1
Step b. gives:	0 0 1	1 1 0	0 1 1	1 0 0
transitional values	0,+1	0,-1	+2,0	-2,0
Step c. gives:	+1	-1	+2	-2

The decimal value of the dual-bit data stream is given by:

$$\sum_{i=0}^{n} b_i \times 2^{2i} \qquad (8)$$

where: $b_i = b_n, b_{n-1}, b_{n-2}, \ldots . b_0$

for the above example:

$$\text{decimal value} = (-2 \times 2^0) + (+2 \times 2^2) + (-1 \times 2^4) + (+1 \times 2^6)$$
$$= -2 + 8 - 16 + 64 = 54$$

With a little practice this encoding procedure can be applied directly to the 2's complement data stream.

The last step of the encoding process is a function of the results obtained from step three and is required to ensure overflows do not occur in the multiplication pipeline. This step is necessary only when a -2 appears in the least significant non-zero position of an encoded multiplicand. In this case the least significant bits of the multiplicand must be modified in a manner which eliminates the -2. In doing so, one must ensure the following bit streams are not introduced in the dual-bit code:

a. * * +2 0 0; the least significant non zero position cannot be a +2.

b. -2 0 -1; a -2 dual-bit slice must be preceded by a +2 or +1.

c. +2 0 +1; a +2 dual-bit slice must be preceded by a -2 or -1.

d. -2 -2 +1; cannot have two successive -2's, they must be separated by +1 or +2.

e. +2 +2 -1; cannot have two successive +2's, they must be separated by -1 or -2.

Examination of Booth's Quaternary algorithm shows that none of the above dual-bit streams will result from the direct application of the algorithm. Therefore, they can only be introduced in the process of eliminating -2's in the least significant non-zero position.

3.2 Signed Two's Complement Arithmetic Using Encoded Multiplicands

In general, serial-pipelined multiplication in signed 2's complement arithmetic with the encoded multiplicands as derived above requires five arithmetic algorithms. These algorithms are based on the dual-bit values of the coded multiplicand (i.e. +1,-1,+2,-2, & 0). The next partial product in each case is obtained as follows:

a. To multiply by +1:

   1. Add multiplier to the input partial product.

   2. Truncate two LSBs of new partial product and fill two most significant positions with sign extensions.

b. To multiply by -1:

   1. Do 2's complement of multiplier by complementing input data stream and adding one.

   2. Add result from step one to input partial product.

   3. Truncate two LSBs of new partial product and fill two most significant positions with sign extensions.

c. To multiply by +2:

   1. Shift multiplier left one position (i.e. truncate sign bit in most significant position and zero fill least significant position). Add result to

input partial product.

2. Truncate two LSBs of new partial product and fill two most significant positions with sign extensions.

d. To multiply by -2:

1. Shift multiplier left one position (i.e. truncate sign bit in most significant position and zero fill least significant position).

2. Take 2's complement of result and add to input partial product.

3. Truncate two LSBs of new partial product and fill two most significant positions with sign extensions.

e. To multiply by 0:

1. Create new partial product by truncating the two least significant positions and sign extending the two most significant positions of the input partial product.

To illustrate the above algorithms and show some additional requirements and characteristics, consider the two examples shown below.

```
example 1: multiplier (MR)    = 0 0 1 1 0 1 1 0 0 = +108
           multiplicand (MC)  =         -1  -2  +1  = -23
           +1xMR/PP0        |0 0|0 0 1 1 0 1 1|0 0
           -2xMR          _____1 0 0 1 0 1 0 0 0|
           PP1            |1 1|1 0 1 0 0 0 0|1 1
           -1xMR          _1 1 0 0 1 0 1 0 0|_____
           FPP             1 0 1 1 0 0 1 0 0|1 1 0 0

           FPP (without truncation) = -2484

example 2: multiplier (MR)    = 0 0 1 1 0 1 1 0 0 = +108
           multiplicand (MC)  =       +2   0  -1  = +31
           -1xMR/PP0        |1 1|1 1 0 0 1 0 1|0 0
            0xMR          _____0 0 0 0 0 0 0 0 0|
           PP1            |1 1|1 1 1 1 0 0 1|0 1
           +2xMR          _0 1 1 0 1 1 0 0 0|_____
           FPP             0 1 1 0 1 0 0 0 1|0 1 0 0

           FPP (without truncation) = +3348
```

First, note that when multiplying by +2 or -2 the last bit of the multiplier is not considered in the calculation of the next partial product. Therefore, all input multiplier data words must have at least two sign bits. Otherwise an overflow would occur when multiplying by +2 or -2. This is not a restriction imposed by the algorithms but necessary for the proper functioning of the +2 & -2 multiplicand modules (see sections 3.3.4 & 3.3.5). Second, the LSBs of the final partial product (FPP) are not truncated and no sign extensions are added. Last, the number of bits in the FPP is always equal to the number of bits in the multiplier.

3.3 Macrocells

Eight macrocells are used in the multiply pipeline. There are five multiplicand cells which perform the +1, -1, +2, -2, and 0 multiplication algorithms discussed in 3.2, two control cells, and one trivial multiplication cell. The five multiplicand cells are called MULTP1, MULTN1, MULTP2, MULTN2, and MULT0. The two control cells are used to locally generate pulsed signals which control the functional state of each multiplicand cell. These cells are called MULTCON and ENDMULTCON.

The designs of the fixed coefficient multiplicand and control modules were derived from Lyon's five-level encoded variable coefficient module presented in [7:422]. The trivial multiplication cell, MULTT0, is simply a series of

three MSFFs. It is used in special situations where the coded multiplicand data word contains all zeroes except in the most significant position and this position is occupied by a +1. All of these macrocells were designed to provide both vertical and horizontal modularity. This allows efficient utilization of silicon area by stacking in both directions. The remainder of this chapter discusses the logic and functional operation of each cell.

3.3.1 Multiplicand Control Cells (MULTCON & ENDMULTCON). The logic diagrams of both the multiplicand control (MULTCON) and positive one (MULTP1) cells are illustrated in Figure 3.1. First, consider the control cell. The same control cell is used by all of the multiplicand modules. As shown in the figure, this cell is composed of three MSFFs, a NAND gate, and eight inverters. The output control signals generated are sign extend (SE), SE bar, reset delay complement (RDC), reset (RST), and RST bar. The sign extend signal is used to control the T-gate, to truncate the two LSBs of the partial product, and sign extend the MSB twice. Reset delay complement is used in the +2 and -2 modules to prevent the MSB of the previous multiplier from effecting the summation of the LSBs in the current calculation. The two reset signals are used to preset the value of the carry-in bit at the beginning of each multiplication process.

All of these signals are produced by passing a control word consisting of all ones and a single zero through the

flip-flops, FFs. This control word has the same bit length as the input multiplier (MR) and the zero is aligned so that it enters the first control FF (i.e. the rightmost FF) at the same time the LSB of the multiplier is entering the multiplicand cell. When the contents of all the control FFs are 1, the value of each output is; RST bar = 1, RST = 0, RDC = 1, SE = 0, and SE bar = 1.

When the zero is propagated through the cell, RST bar, RST and RDC transition to the complement of their normal state for one clock cycle. And, SE and SE bar are complement for two clock cycles. Also, note that the transitions of RST and RST bar occur on the rise of phi2, while the transitions of the remaining signals occur on the rise of phi1. This enables the carry in flip-flops in the multiplier cells to be reset on phi2 so that the least significant carry-in bit arrives at the adder on the rise of phi1.

The only difference between the standard control cell described above and the end control cell (ENDMULTCON) is that SE and SE bar are hardwired to their normal states. This cell is used to control the multiplicand cell in the most significant dual-bit position of the multiplier pipeline.

3.3.2 Positive One Multiplicand Cell (MULTP1). The functional operation and hardware circuit for the various multiplicand cells are similar in many aspects. The basic

differences between the cells are the specific operations performed on the multiplier bit stream before it is summed with the incoming partial product (PPI), and the initial value of the carry-in bit (CYI). This section presents a detailed explanation of the positive one cell which also forms the basic structure and functionality of the -1, +2, and -2 modules.

The positive one multiplicand module is composed of five MSFFs, a full adder and a transmission gate. These components are interconnected to form two paths through the cell (i.e. multiplier input to multiplier output and PPI to PPO) and a feedback path between the adder circuit. The multiplier path is formed by three MSFFs across the top of the cell and the PP path goes through the adder and one MSFF. The carry feedback path has one MSF.

Data flow through the module is synchronized by the dual-phase clocks, phi1 and phi2. Initialization and control of the cell is achieved by three signals, SE, SE bar, and RST (note, RST bar and RDC are not used in this cell). Both data words (i.e multiplier and PPI) are input to the module serially, LSB first. All of the multiplier cells use the same control module.

The following example illustrates the operation of the MULPT1 and MULTCON modules. For assistance in keeping track of the current states of the modules, during the following example, the reader is referred to Figure 3.1 and Table 3.2.

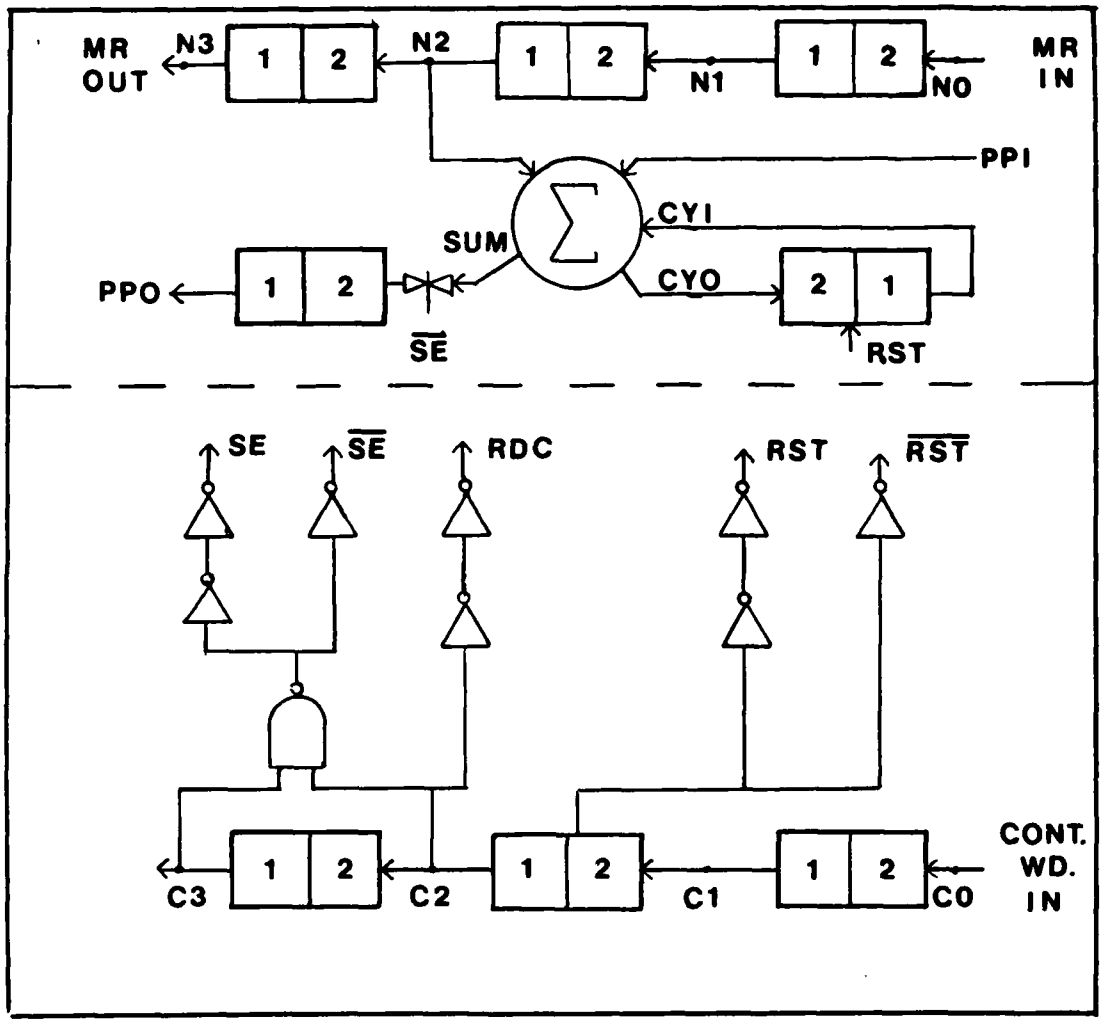


Figure 3.1 Macrocells (a) MULTP1 (b) MULTCONT

MR = 1101011, PPI = 1010100, Cont.Wd = 0111111011..

NODE	Status of Cell After Clock Cycle XX											
	11	10	9	8	7	6	5	4	3	2	1	0
C1	1	1	1	0	1	1	1	1	1	1	0	1
C2	1	1	0	1	1	1	1	1	1	0	1	1
C3	1	0	1	1	1	1	1	1	0	1	1	1
CYO	*	*	*	1	0	0	0	0	0	0	*	*
CYI	*	*	"0"	0	0	0	0	0	0	"0"	*	*
N2	Z2	Z1	Z0	1	1	0	1	0	1	1	Y6	Y5
PPI	P2	P1	P0	1	0	1	0	1	0	0	P6	P5
SUM	S2	S1	S0	0	1	1	1	1	1	1	S6	S5
PPO	0	0	0	1	1	1	1	S6	S6	S6	S5	S4
							↑ LSB					

Table 3.2 Sample Data Flow for MULTP1

-First, assume that at clock cycle 0:

--the LSB of the input multiplier (MR), X_n = 1101011 is at the N0 input of the module and the two MSBs of the previous multiplier, Y5 and Y6, are at nodes N2 and N1 respectively.

--the PPI = P5 P6 0 0 1 0 1 0 1, where P5 and P6 are the two MSBs of the partial product results from a preceding module that are to be added to Y5 and Y6. And the next seven bits are those to be bit-serially added to the next MR input. Notice that "relative to the cell boundary", the LSB of the PPI is always two clock cycles behind the LSB of the input MR data word.

--and last, the input to the control module is given by 0 1 1 1 1 1 1 0 1 etc. Note that the control word is aligned so that the zero enters at the same time the LSB of the multiplier enters.

-Now, as the clocks are cycled (i.e. one cycle consist of two non-overlapping clock pulses, a phi2 pulse followed by a phi1 pulse) the two modules function as follows:

--after CC-0 (i.e. the fall of phi1) and before the rise of phi2 in CC-1:

---the sum (S5) and carry out (CYO=*) of P5, Y5 and the current carry in (CYI), is produced by the adder.

--on the rise of phi1 (CC-1):

---the zero control pulse for the multiplier word (MR) X_n is shifted to node C1.

---the CYO bit is loaded into the CYI storage register.

---S5 is shifted to the partial product out register (FF1).

---P6 and Y6 are shifted to the inputs of the adder.

--after CC-1 and before the rise of phi2 for CC-2:

---S6 (P6 + Y6 + CYI) and the new CYO are produced by the adder.

--during CC-2:

---the control zero propagates from C1 to C2 causing the control signal RST to pulse high on the rise of phi2 and thus reset the CYI register to 0.

---the most significant bit of the previous summation, S6, is shifted to the PPO FF on the rise of phi2.

---on the rise of phi1; the CYI bit, 0, and the LSBs of the X_n multiplier , 1, and PPI, 0, are shifted to the inputs of the adder.

--after CC-2 and before CC-3 :

---during the time period between the fall of phi1, CC-2, and the rise of phi2, CC-3, - the adder computes the least significant SUM bit, 1, and the CYO bit, 0.

---the control zero at C2 sets SE bar low, thus blocking the flow of data through the transmission gate.

--during CC-3:

---the control zero propagates from C2 to C3.

---S6 is retained in the PPO FF, thus providing a sign extension for the partial product out of the previous multiply operation (i.e 1 x Y_n). In doing so, the LSB of the PPO for 1 x X_n is truncated.

---the CYO bit,0, is shifted to the CYI register.

---the next two LSBs of the MR, 1, and PPI, 0, are shifted to the inputs of the adder.

--after CC-3 and before CC-4 :

---the adder computes the next SUM and CYO bits, 1 & 0.

---the control zero at C3 holds SE bar low, thus blocking the flow of data through the transmission gate.

--during CC-4:

---the control zero propagates out of the control cell.

---S6 is retained in FF1, thus providing a second

sign extension for the partial product out of the previous multiply operation. As before, the next LSB of the PPO for 1 x X_n is truncated.

---the CYO bit, 0, is shifted to the CYI register.

---and the next two LSBs of the MR, 0, and PPI, 1, are shifted to the inputs of the adder.

--after CC-4 and before CC-5 :

---the adder computes the next SUM and CYO bits, 1 & 0.

--during CC-5:

---the CYO bit, 0, is shifted to the CYI register.

---the first LSB of the PPO for 1 x X_n is shifted into FF1.

---and the next two LSBs of the MR, 1, and PPI, 0, are shifted to the inputs of the adder.

This process continues in the manner described above until the control zero enters the module at CC-8. At this point the modules begin to re-initialize (i.e. reset CYI, sign extend MSBs, and truncate LSBs of new PPO) to determine the PPO for the next multiplication. As can be seen from this example, the positive one multiplicand module implements the arithmetic algorithm discussed in section 3.2 of this report.

3.3.3 Negative One Multiplicand Cell (MULTN1). There are only two hardware differences between the positive and negative one multiplicand cells (see Figure 3.2). First, the negative one cell has an inverter between node N1 and the input to its adder. And second, the carry-in (CI) storage register is reset to 1 (as opposed to 0 in the

negative one cell) via RST bar. These changes are required to produce the 2's complement of the input multiplier bits before adding them to the input partial product.

As shown in the sample calculation summarized in table 3.3, when the control 0 propagates from C1 to C2, during clock cycle 2, the CYI register is reset to 1. In addition, the LSB of the multiplier is complemented and transmitted to the adder. In subsequent cycles, the complement of the next MR bit is summed with the next PPI bit and the CYI bit from the previous addition. Therefore, initially setting the CYI bit to 1 and complementing, each MR bit performs the 2's complement operation. The bit-serial addition of the 2's complemented MR to the PPI calculates the PPO.

Excluding the differences discussed above, the functional operation of the negative one cell is identical to the positive one cell. The two LSBs of the partial product out are truncated and two sign extensions are appended in the most significant positions.

3.3.4 Positive Two Multiplicand Cell (MULTP2). The algorithm used to multiply by a +2 calls for shifting the MR data word left one bit position, truncating the MSB and filling the least significant position with 0. Then this result is added to the PPI. In the MULTP2 cell, inputting the multiplier bits from node N3 effectively shifts the multiplier data word left one position (in time) relative to the partial product in. Also, resetting RDC low one clock cycle prior to the arrival of the least significant

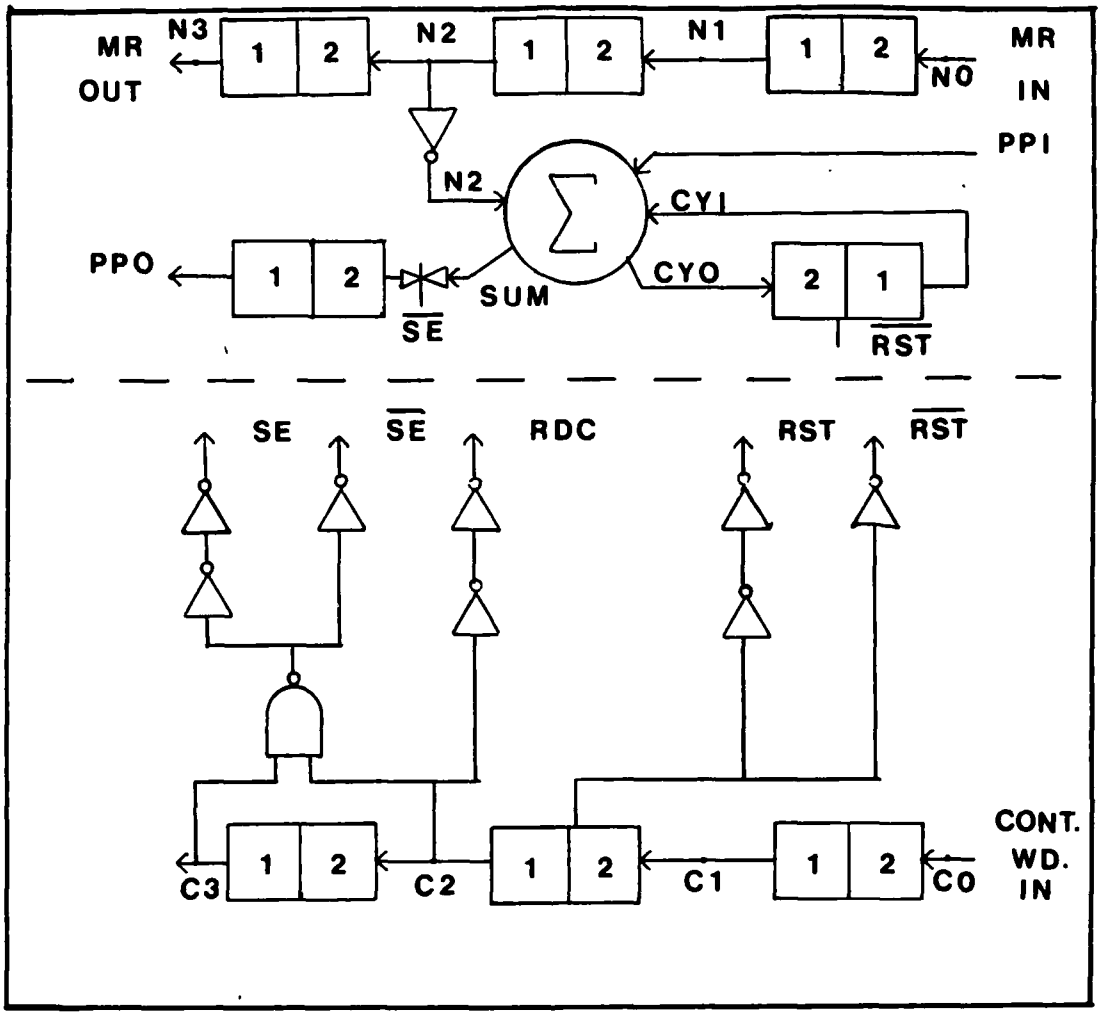


Figure 3.2 Macrocells (a) MULTN1 (b) MULTCONT

MR = 1101011, PPI = 1010100, Cont.Wd = 0111111011..

NODE	Status of Cell After lock Cycle XX											
	11	10	9	8	7	6	5	4	3	2	1	0
C1	1	1	1	0	1	1	1	1	1	1	0	1
C2	1	1	0	1	1	1	1	1	1	0	1	1
C3	1	0	1	1	1	1	1	1	0	1	1	1
CYO	*	*	*	0	0	1	0	1	0	0	*	*
CYI	*	*	"1"	0	1	0	1	0	0	"1"	*	*
N2	Z2	Z1	Z0	0	0	1	0	1	0	0	Y6	Y5
PPI	P2	P1	P0	1	0	1	0	1	0	0	P6	P5
SUM	S2	S1	S0	1	1	0	1	0	0	1	S6	S5
PPO	1	1	1	1	0	1	0	S6	S6	S6	S5	S4
							↖LSB					

Table 3.3 Sample Data Flow for MULTN1

multiplier bit will cause a 0 to be input to the adder and subsequently summed with the LSB of the PPI and the CYI bit which is also reset to 0. These logical and control actions combine to do the multiply by +2 algorithm.

The hardware design of the positive two multiplicand cell is depicted in Figure 3.3. As illustrated, the only difference between it and the positive one cell is the manner in which the multiplier bits are transmitted into the adder. In this module the multiplier bits are input from node N3 where they are logically ANDed with the control signal RDC and then propagated into the adder. Notice that when RDC is low, a 0 is input to the adder and when RDC is high, the value of the multiplier bit is not changed.

The example summarized in Table 3.4 shows that the control zero propagating from C1 to C2 causes RST to go low and reset the CYI register to 0. Also, with C2 reset low RDC will be low. This causes a 0 to be input to the adder at node N3'. The add, shift, truncate, and sign extend operations from CC-3 through CC-8 are as previously described. Then, at CC-9, initialization begins for the computation of the next PPO. Notice that in this cell the MSB of the multiplier is not considered in the computation of the PPO. As such, it is essential that there be at least two sign bits in the multiplier data word for the cell to function properly.

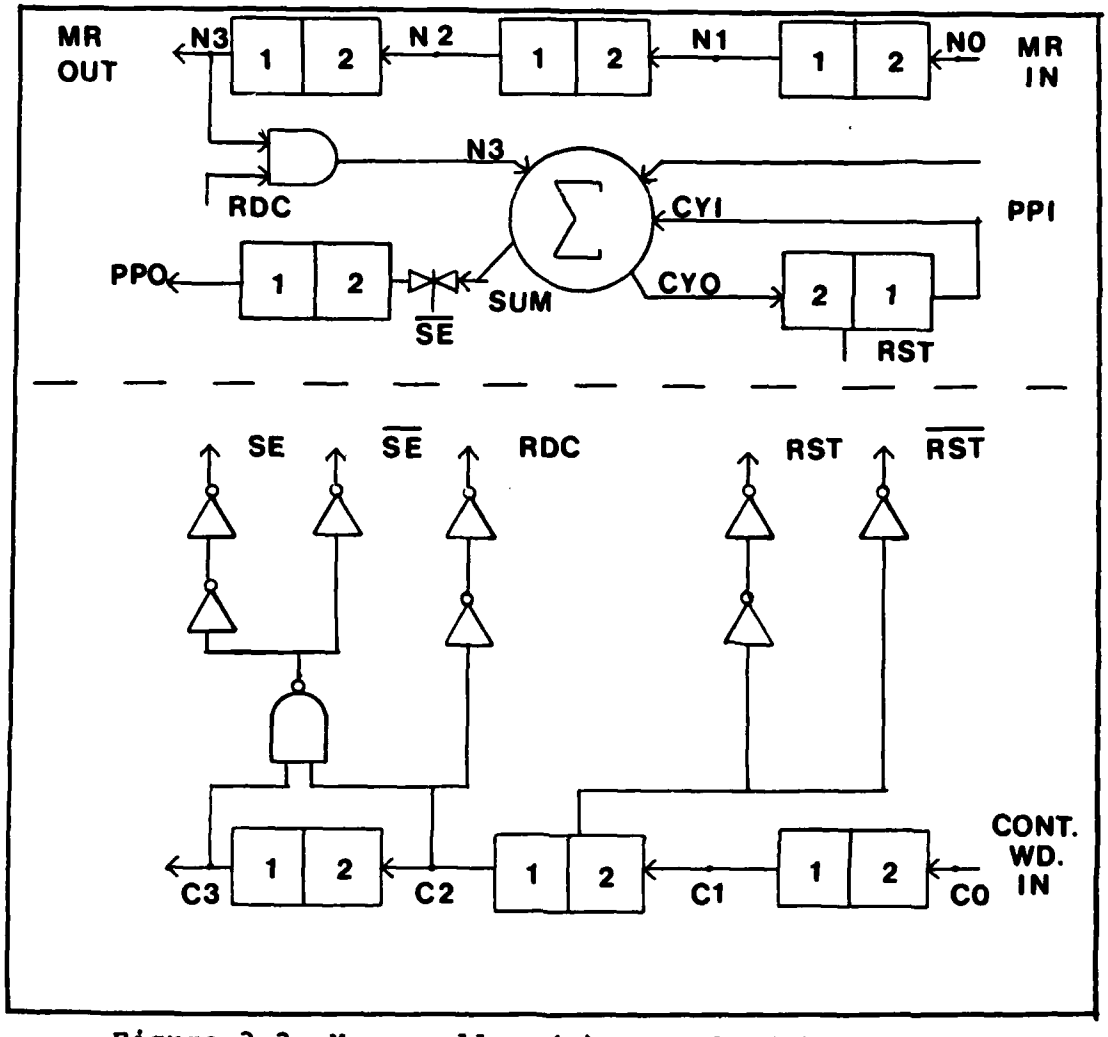


Figure 3.3 Macrocells (a) MULTP2 (b) MULTCONT

MR = 1110101, PPI = 0001011, Cont.Wd = 0111111011.

NODE	Status After Clock Cycle XX											
	11	10	9	8	7	6	5	4	3	2	1	0
C1	1	1	1	0	1	1	1	1	1	1	0	1
C2	1	1	0	1	1	1	1	1	1	0	1	1
C3	1	0	1	1	1	1	1	1	0	1	1	1
CYO	*	*	*	0	0	0	1	0	1	0	*	*
CYI	*	*	"0"	0	0	1	0	1	0	"0"	*	*
N3'	Z2	Z1	"0"	1	1	0	1	0	1	"0"	Y6	Y5
PPI	P2	P1	P0	0	0	0	1	0	1	1	P6	P5
SUM	S2	S1	S0	1	1	1	0	1	0	1	S6	S5
PPO	1	1	1	1	1	0	1	S6	S6	S6	S5	S4
							↑ LSB					

Table 3.4 Sample Data Flow for MULTP2

3.3.5 Negative Two Multiplicand Cell (MULTN2). The objective of the negative two multiplicand cell is to multiply the input data word by -2 and bit-serially add this result to the PPI to form a new PPO. The arithmetic algorithm used to multiply by -2 requires two functional steps. First, the MR data word is shifted left one position with the MSB being truncated and the least significant position filled with 0. This is accomplished in the same way as in the +2 module. A 1 is input to the adder (by resetting RDC low) to be summed with the LSB of the PPI and the CYI which is set to 1 by resetting RST bar low. Note that inputting a 1 to the adder at N3' is equivalent to inputting a 0 at N3. Second, the 2's complement of the shifted MR data word is added bit-serially to the PPI. To obtain the 2's complement, each MR data bit input to the adder is inverted and a 1 is added in the least significant position by setting CYI to 1 as described above.

The module's hardware design is illustrated in Figure 3.4. The multiplier bits are input at node N3 where they are logically ANDed with the control signal RDC. This logical result is then inverted and propagated to the adder input, N3'. When RDC is low, a 1 is input to the adder and when RDC is high, the complement of the multiplier bits are input to the adder.

To further illustrate these functions, consider the example shown in Table 3.5. When the control zero propagates from C1 to C2, RST bar is reset low, thereby

causing the CYI register to be set high. Additionally, node C2 is low which causes a 1 to be input to the adder. Between clock cycles 2 and 3 the adder computes the LSB of the PPO which, as described above, is the sum of:

a. the LSB of the PPI.

b. the 0 filled in the LSB of the MR data word which is inverted to a 1 before being input to the adder.

c. and the CYI bit which is equal to 1.

Except for complementing each multiplier bit, the add, shift, truncate, and sign extend operations from CC-3 through CC-8 are as previously described in section 3.3.2. At CC-9 the initialization process for the next multiplication starts. Also note that this cell requires two sign bits to operate properly because the MSB of the multiplier is not considered in the computation of the PPO.

3.3.6 Zero Multiplicand Cell (MULT0). When a dual-bit position of the multiplicand data word is occupied by a 0, the new PPO is formed by simply truncating the two LSBs of the PPI and sign extending the MSB twice. Also, the MR data word is delayed three clock cycles to maintain its relative position within the pipeline. As illustrated in Figure 3.5(a), the hardware used to accomplish these functions is composed of four MSFFs and a transmission gate. The MR data word is delayed three cycles by the FFs across the top of the cell. Truncation and sign extending is performed by

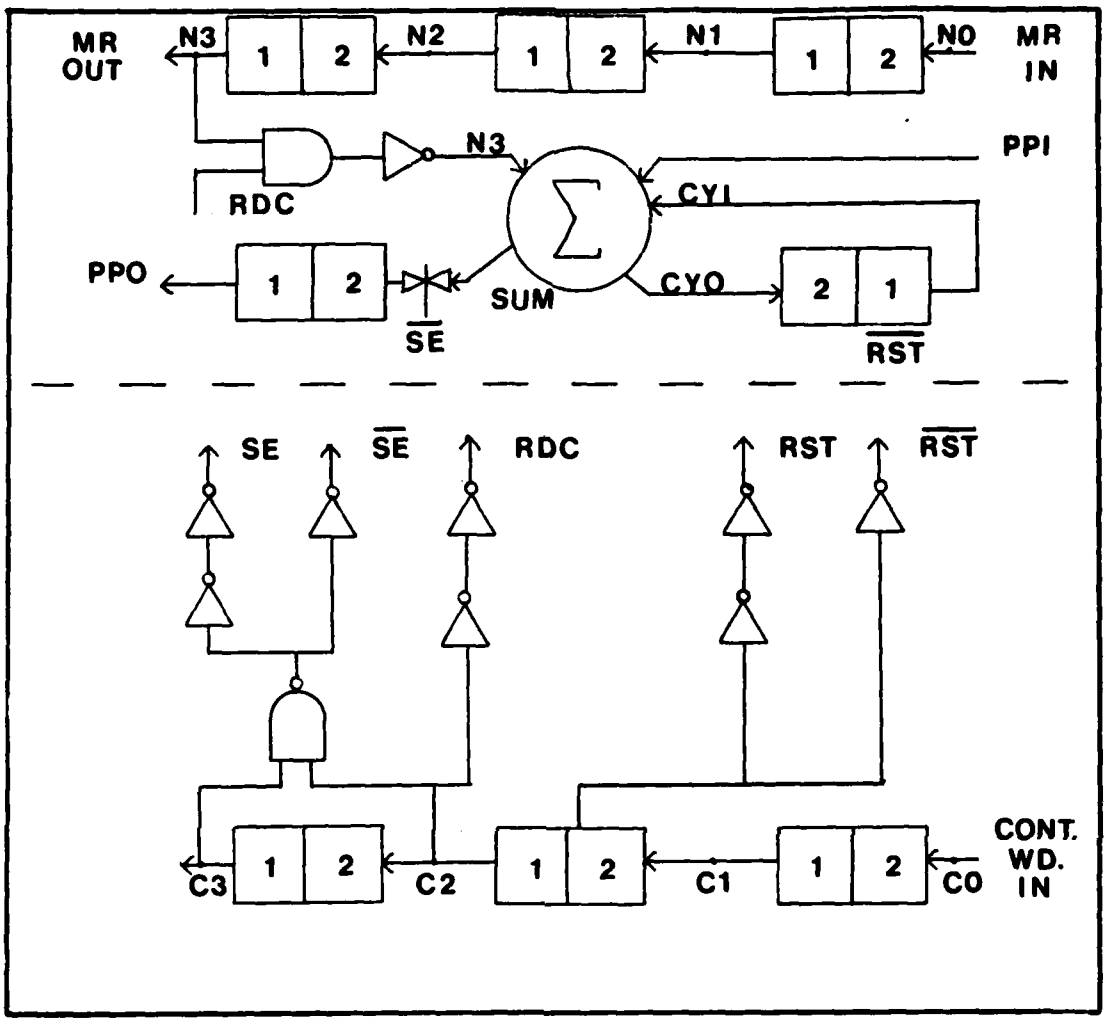


Figure 3.4 Macrocells (a) MULTN2 (b) MULTCONT

MR = 0001110, PPI = 0010101, Cont.Wd = 0111111011..

NODE	Status After Clock Cycle XX											
	11	10	9	8	7	6	5	4	3	2	1	0
C1	1	1	1	0	1	1	1	1	1	1	0	1
C2	1	1	0	1	1	1	1	1	1	0	1	1
C3	1	0	1	1	1	1	1	1	0	1	1	1
CYO	*	*	*	0	0	0	0	1	1	1	*	*
CYI	*	*	"1"	0	0	0	1	1	1	"1"	*	*
N3'	Z2	Z1	"1"	1	1	0	0	0	1	"1"	Y6	Y5
PPI	P2	P1	P0	0	0	1	0	1	0	1	P6	P5
SUM	S2	S1	S0	1	1	1	1	0	0	1	S6	S5
PPO	1	1	1	1	1	1	0	S6	S6	S6	S5	S4
							↑LSB					

Table 3.5 Sample Data Flow for MULTN2

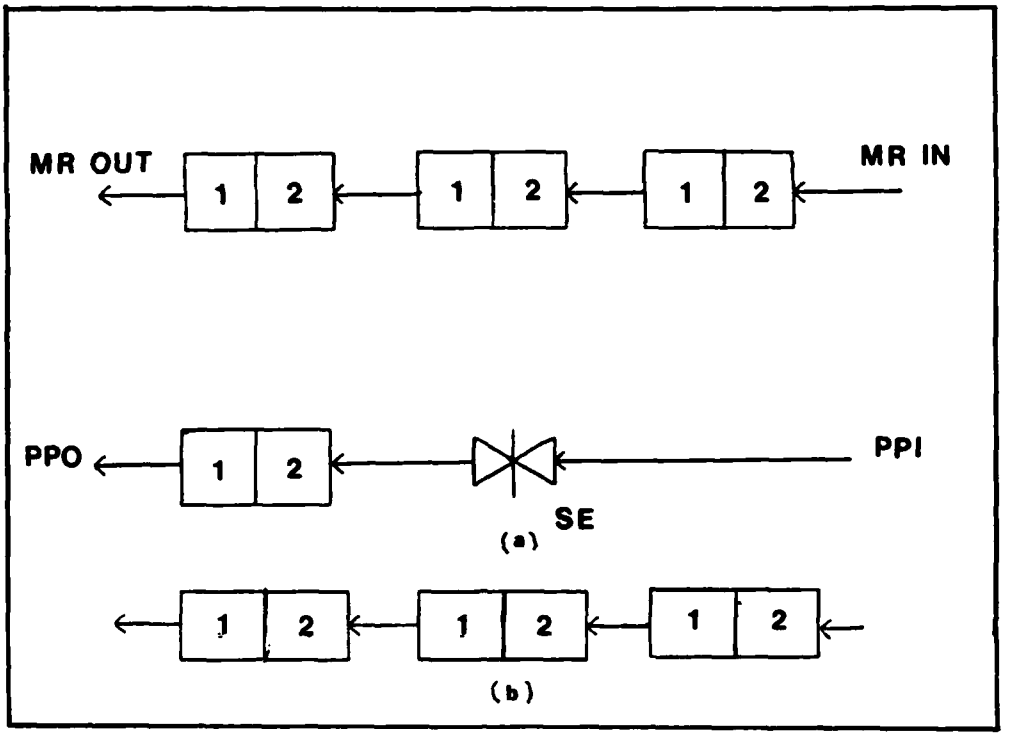


Figure 3.5 Macrocells (a) MULT0 (b) MULTT

resetting SE bar low for two cycles coinciding with the arrival the two LSBs of the PPI and the departure of the MSB of the PPO. Control of SE bar is provided by the MULTCON cell.

3.3.7 Trivial Multiplicand Cell (MULTT). This is a special cell designed to do trivial multiplications which occur when the MSB of the dual-bit multiplicand word is a 1 and the remaining positions are 0. It consists of three MSFFs, see Figure 3.5 (b), which are used to delay the propagation of the multiplier data word three cycles for each dual-bit position of the multiplicand. The rationale for using this scheme to do trivial multiplication is to reduce the number of devices in the multiplier pipeline, thereby reducing the circuit's power consumption, capacitive loading, and silicon area.

3.4 Formation and Analysis of Multiplicand Pipeline

An 8 x 8 serial multiplier pipeline is constructed by connecting multiplicand cells in accordance with the order of the coded dual-bit constant coefficient. For example, if the encoded coefficient is given by -1 +2 -2 +1, then the pipeline would be assembled as shown in Figure 3.6. The multiplier data word is input LSB first and is serially passed from the least significant dual-bit slice (2^0) to the most significant dual-bit slice (2^4). As the MR data word is clocked through the pipeline, each cell computes an output partial product and passes it to the PPI of the next higher order cell. The final partial product output is the result of the multiplication process.

The control word for an eight-bit multiplier would be 01111111 and would be input to the control modules so that the 0 is synchronized with the LSB of the multiplier. Excluding the highest order dual-bit slice, standard control modules (MULTCON) are used throughout the pipeline. Since the partial product out of the highest order multiplicand module is the final output, no truncations or sign extensions are required. Therefore, the ENDMULTCON module is used to control the functional operation of the most significant multiplicand cell. This not only precludes the truncation of valid information, but also provides a mechanism for the final product bit stream to exit the pipeline with its associated multiplier bit stream.

The partial product input to the least significant

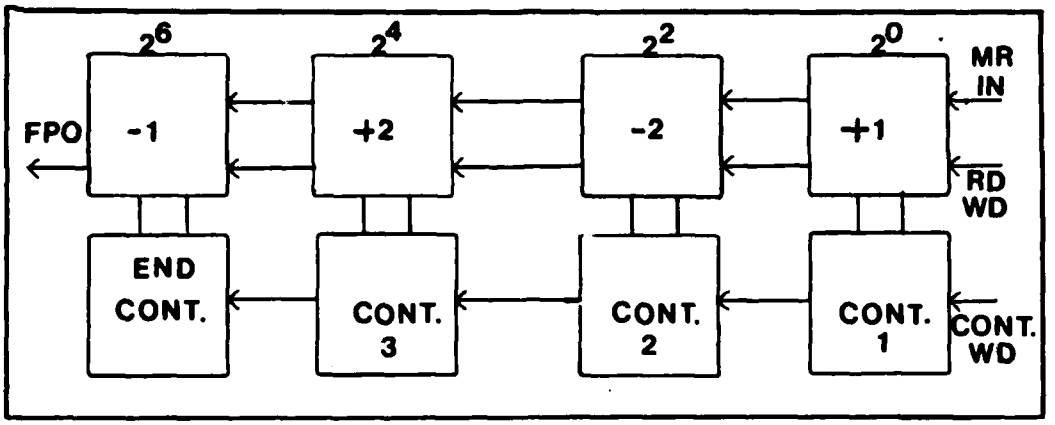


Figure 3.6 Sample Dual Bit Multiplier

multiplicand cell can be tied to ground or used to increase the accuracy of the final product by performing a rounding operation. To achieve rounding a predetermined control is input to PPI terminal. This word contains only one "logical one" bit and its length is equal to the MR bit length. The relative position, X_n (n = 1,2,3,...), of the logical one is a function of the multiplicand dual-bit length and is given by: n = 2(m -1) where m is the number of dual-bit positions in the coded multiplicand. To further illustrate the pipeline's operation, consider the following sample calculation:

Sample calculation: Using the pipeline shown in Figure 3.6, determine the final output product if the input multiplier is given by 00010101 and the rounding function is used.

a. The number of dual-bit positions, m, is four. Therefore, n = 6 and the rounding word is given by 00100000.
b. Calculate final output product using the algorithms for each multiplicand cell:

```
                                0 0 0 1 0 1 0 1   (MR)
                                -1  +2  -2  +1    (MC)
Rounding Word                   0 0 1 0 0 0 0 0
                                0 0 0 1 0 1 0 1   +1xMR
                          0 0 0 0 1 1 0 1 0 1     PP0
                          1 1 0 1 0 1 1 0         -2xMR
                    1 1 1 1 1 1 0 0 1 1           PP1
                    0 0 1 0 1 0 1 0               +2xMR
              0 0 0 0 1 0 0 1 1 0                 PP2
              1 1 1 0 1 0 1 1                     -1xMR
              1 1 1 1 0 1 0 0                     Final PP
```

To accomplish several multiplications simultaneously, all one need do is to stack the number of pipelined multipliers required and simultaneously input the MR data words, rounding words, and control word.

3.4.1 Overflow Condition. The example illustrated below demonstrates how overflow can occur if the multiplier bit stream into the pipeline has less than two sign bits.

```
                                0 1 0 1 0 1 0 1   (MR)
                                -1  +2  -2  +1    (MC)
Rounding Word                   0 0|1 0 0 0 0 0
                                0 1|0 1 0 1 0 1   +1xMR
                          0 0 0 1|1 1 0 1 0 1     PP0
Overflow                  0 1 0 1|0 1 1 0         -2xMR
                    0 0 0 1 1 1|0 0 1 1           PP1
                    1 0 1 0 1 0|1 0               +2xMR
              1 1 1 1 0 0 0 1|1 0                 PP2
              1 0 1 0 1 0 1 1|                    -1xMR
              1 0 0 1 1 1 0 0|                    Final PP
```

This hazard is independent of the overflow conditions discussed in section 3.1. Therefore, all multiplier inputs in the pipeline must have a least two sign bits.

3.5 Macrocell Sizes and Density

Table 3.6 summarizes the macrocell sizes and density. The size shown is the array size. All of the cells are designed to stack vertically and horizontally without having to add additional interconnects (i.e. vias, contacts, or metal runs).

Cell Name	Arrayed Size (λ)	# Devices/cm_2
MULTP1	263x86	112,500
MULTN1	263x86	126,300
MULTP2	263x86	128,200
MULTN2	263x86	130,100
MULT0	263x86	68,900
MULTT	263x46	84,000
MULTCONT	263x80	86,400

Table 3.6 Dimension/Density of Multiplier Macrocells (3 micron design rules)

3.6 SPICE Simulation Results

The worst case propagation delay in the pipeline occurs in the multiplier array. The specific location of this delay is in the column that contains the most negative two modules (MULTN2). SPICE simulations of the sum and carry-out signals in these modules show that the worst case rise/fall times are:

a. Using 0 to 90% measure - approximately 7.5 ns

b. Using 50 to 50% measure - approximately 4.5 ns

These simulations were based on the MOSIS three micron processing technology. Therefore, when scaled down to a one and one half micron technology, the arithmetic circuitry will operate above 80 MHZ.

The circuit modeled by these simulations included; (1) the full adder, (2) phil latch of carry-in flip-flop, (3) phil latch of partial product input, (4) phil latch/NAND input for the multiplier bits, (4) the phi latch/NAND gate input from the multiplier controller cell (i.e. RDC signal), and (5) the additional gate and metal line capacitance loading down the RDC control line.

3.7 Multiplier Pipeline for 16-Point WFT

3.7.1 General Characteristics and Limitations. The coded multiplicands for the 16-point WFT coefficients are summarized in Table 3.7. The decimal values in the left column are the results obtained after scaling the original coefficients down by a factor of two.

After applying Booth's Quaternary algorithm, the dual-bit multiplicand codes for MC8 and MC16 were modified to eliminate the negative two in their least significant positions. Since these modifications were in the 30th bit position, the noise introduced is on the order of 2^{-30} and will have only a minimal effect, if any at all, on the numerical precision of the 16-point transform.

Scaled Decimal Coefficient	Rounded 28-bit Binary Conversion and Dual-Bit Code
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC01	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC02	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC03	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC04	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC05	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
.3535533905	0 0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1
MC06	+1 | -1 | -1 | +1 | +1 | 0 | 0 | +1 | +1 | 0 | -1 | +1 | -1 | +1
.3535533905	0 0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 1 0 1 1
MC07	+1 | -1 | -1 | +1 | +1 | 0 | 0 | +1 | +1 | 0 | -1 | +1 | -1 | +1
.1913417165	0 0 0 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0
MC08	0 | +2 | -2 | 0 | +2 | 0 | -1 | +2 | 0 | -2 | +1 | -1 | -1 | -2
Modified to —	0 | +2 | -2 | 0 | +2 | 0 | -1 | +2 | 0 | -2 | +1 | -1 | -1 | -1
.6532814825	0 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 1 0 0 1 0
MC09	+1 | +1 | +1 | 0 | -2 | +2 | 0 | -1 | -1 | 0 | -1 | -2 | +1 | 0
-.2705980500	1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 0 0 1 1
MC10	-1 | +2 | -1 | +1 | +1 | +2 | -1 | +1 | 0 | +1 | -1 | -1 | 0 | -1
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC11	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
-.5000000000	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC12	-1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
.5000000000	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
MC13	+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0
-.3535533905	1 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 1
MC14	-1 | +1 | +1 | -1 | -1 | 0 | 0 | -1 | -1 | 0 | +1 | -1 | +1 | -1
-.3535533905	1 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 1
MC15	-1 | +1 | +1 | -1 | -1 | 0 | 0 | -1 | -1 | 0 | +1 | -1 | +1 | -1
-.4619397665	1 1 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 0 0 1 1
MC16	-1 | 0 | +1 | +1 | -1 | +2 | 0 | -1 | +1 | -1 | -2 | 0 | +2 | -2
Modified to —	-1 | 0 | +1 | +1 | -1 | +2 | 0 | -1 | +1 | -1 | -2 | 0 | +2 | -1
.2705980500	0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 0 1
MC17	+1 | -2 | +1 | -1 | -1 | -2 | +1 | -1 | 0 | -1 | +1 | +1 | 0 | -1
-.6532814825	1 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 1 0 1 1 0 1 0 0 1 1 0 1 1 0
MC18	-1 | -1 | -1 | 0 | +2 | -2 | 0 | +1 | +1 | 0 | +1 | +2 | -1 | 0

Table 3.7 Coefficient Coding for 16-Point WFT

The specific output product characteristics and limitations of the pipeline are directly related to the range and format of the input multiplier words. This information is easily obtained by examining the functions performed at each stage of the overall WFT pipeline to determine their effect on the input data points. Figure 3.7 shows the range and word format of the input data points at each stage of the pipeline.

Beginning at the input stage, the input data points are interpreted as being greater than or equal to -1 and less than +1. The data words have a 24 bit format. Bit 24 is a parity bit, and bits 1-23 are data bits containing at least one sign bit in location 23. The binary point is between bits 22 and 23. Next, each data word is extended to produce a 32-bit word format in the same range. As discussed in Chapter 5, sign extensions and zero fills are added as a function of the input scale factor. However, its operation is such that it insures there is a minimum of four sign extensions (bits 28-32) and the binary point is between bits 27 and 28. These 32-bit data words are then passed into the pre-addition stage where three successive levels of addition are possible. Depending on the relative magnitudes of the input data words, each level may absorb one sign bit. Also, successive additions of the largest possible magnitudes increase the potential output range to $-8 \leq R_x < +8$. Therefore, the input to the multiplier pipeline will be a 32-bit word with at least two sign bits

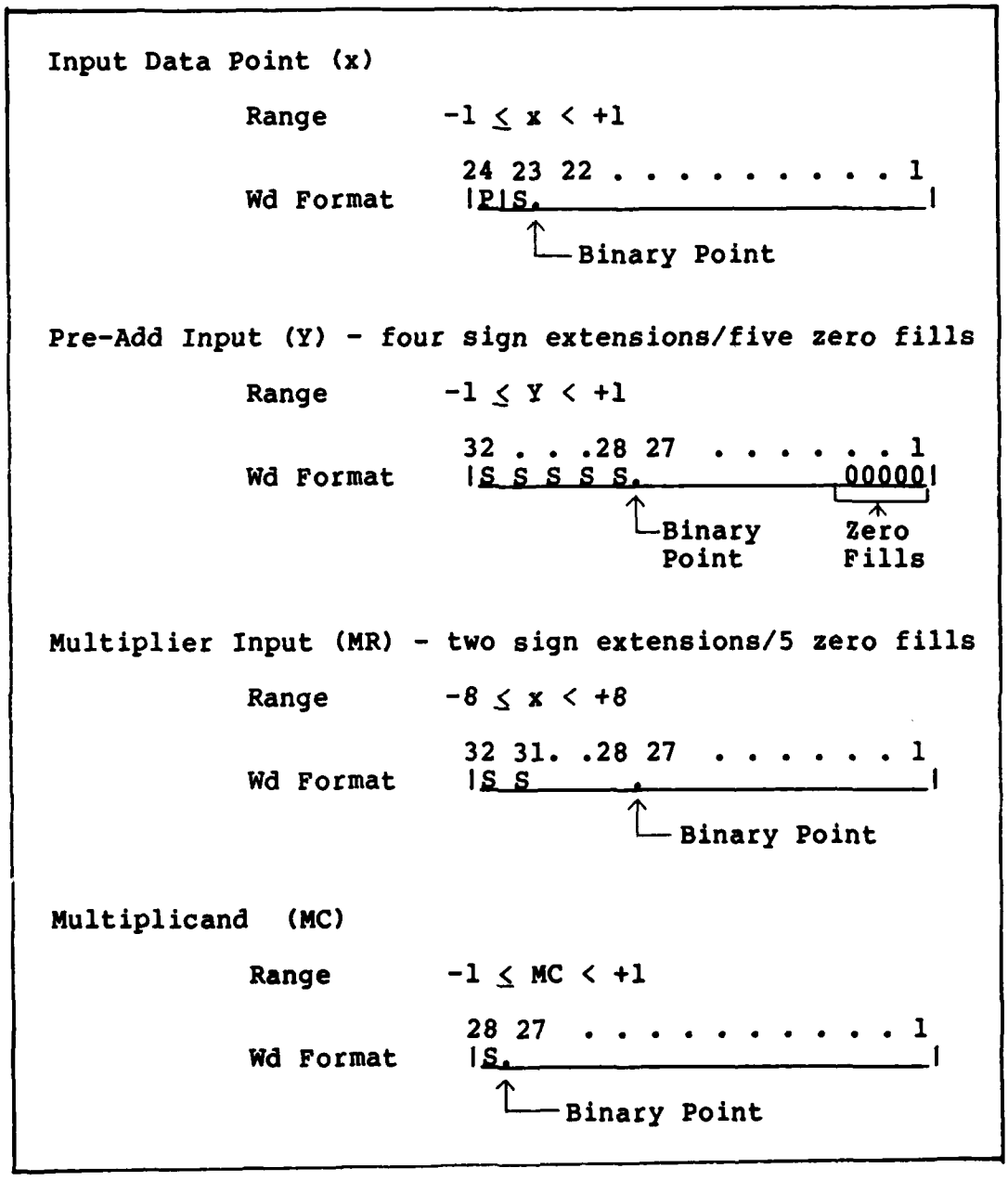


Figure 3.7 Range/Wd Format of 16-Point WFT Processor

(31-32) and its radix will lie between bits 27 and 28.

Given this input format and range, coupled with the range and format of the coded coefficients, several characteristics of the multiplier pipeline can be determined. With a 32-bit multiplier and 28-bit multiplicand, the full range product can require at most 60 bits (see Figure 3.8). There are 27 bits to the right of the radix point in each data word. Therefore, the radix in the resulting 60-bit product will be between bits 54 and 55.

The pipelined multiplication architecture only provides 32 product bits. To address the question of which of the 60 possible bits are kept, one must consider the operation of each dual-bit slice cell in the multiplicand pipeline. As discussed in section 3.3, each dual-bit slice, excluding the most significant, truncates two LSBs of the PPO. Thus, the 26 LSBs are discarded. It can be concluded that the 32-bit product out of the pipeline will consist of bits 27 through 58 of the potential full range product. Relative to the 32 bit product out, the binary point will be between bits 28 and 29. So the outputs are greater than or equal to -8 and less than +8.

Why can bits 59 and 60 be discarded? The rationale for not considering these two bits is directly related to the possible range of multiplier inputs and the pipeline multiplicands. As previously shown, the multiplier inputs will be $-8 \leq MR_X < +8$ and the multiplicands are scaled

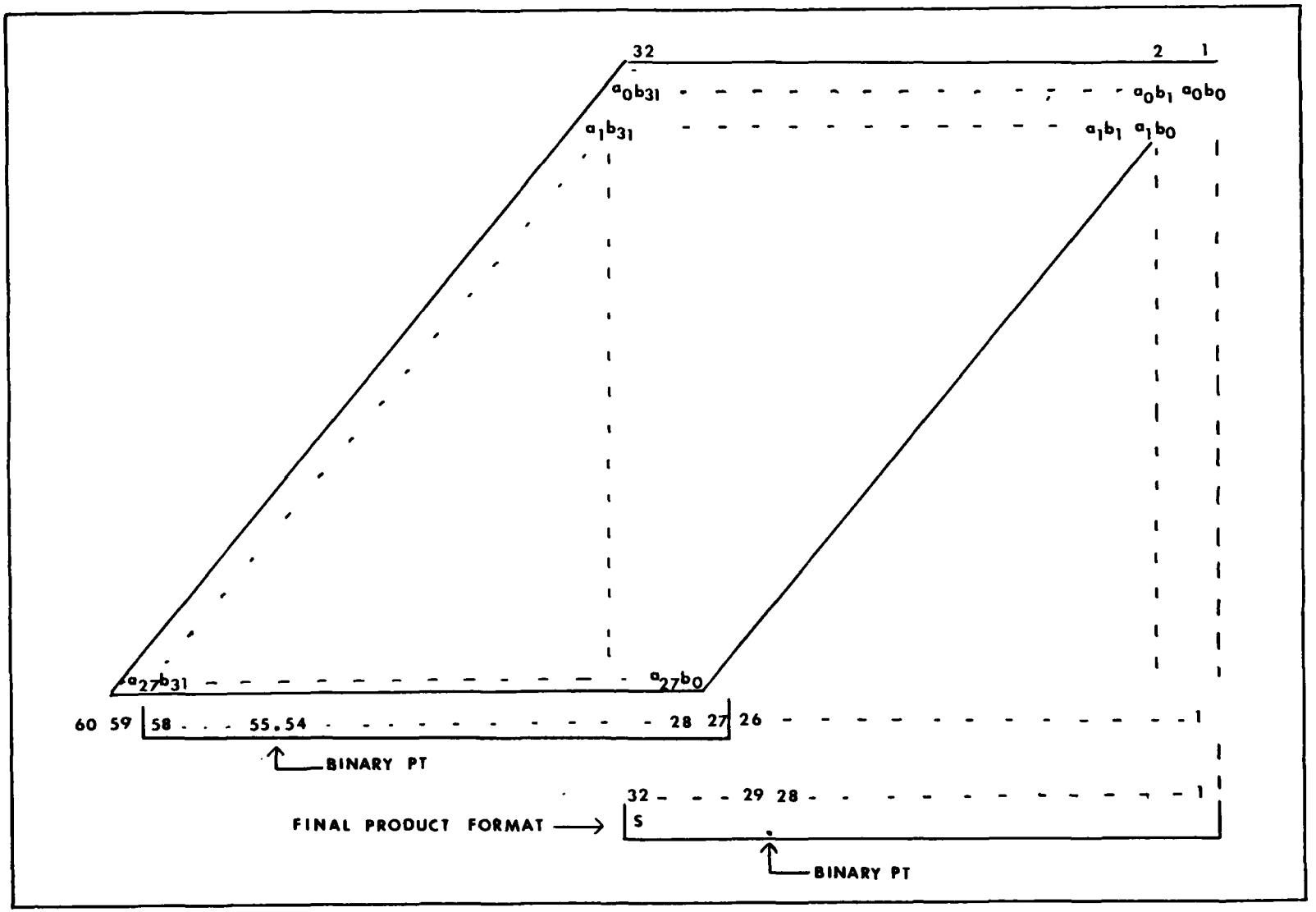


Figure 3.8. Multiplication Parallelogram

down to fall in the range $-1 \leq MCYY < 1$. With these bounds, the magnitude of the largest possible negative product would be less than eight (i.e. -8 x +MCYY, where +MCYY is the largest allowable positive value for the multiplicand, but is less than one). And, the magnitude of the largest possible positive product will be less than eight (i.e. -8 x -MCYY, where MCYY is the largest allowable negative value for the multiplicand, but is greater than negative one). With the radix between bits 28 and 29 of the final product, the range of representable results are numbers greater than or equal to -8 but less than +8. It is seen that bits 59 and 60 would only be sign extensions and are therefore not required to properly represent the final products calculated by the multiplier pipeline.

As discussed in section 3.4, the format of the rounding word input to the PPI of the least significant dual-bit slice is a function of the multiplier bit length and the number of dual-bit slices in the multiplicand. For the 16-point multiplier pipeline, the rounding word contains 32 bits with the one in bit position 26.

3.7.2 Pipeline Optimization. Examination of the coded dual-bit multiplicands reveals that there are eight multiplicands where the trivial multiplicand cell (see section 3.3.7) can be employed. To justify the use of this cell, consider the case where the standard multiplicand cells are used. In the most significant position of the pipeline there would be a MULTP1 cell and the remaining

thirteen positions would be filled with MULT0 cells. In the thirteen least significant positions the multiplier input is propagated through each cell unchanged. But, the rounding word's 26 LSBs are truncated (i.e 25 zeroes and the rounding bit) and replaced with 26 sign extensions, zero fills. Therefore, the inputs to MULTP1 will be a PPI that consists of all zeroes and the original multiplier word.

Since the last cell in the pipeline does not do truncations or sign extensions, the product out will simply be the original input multiplier bit stream. The only difference between the product out and the multiplier in is a matter of where the radix falls. As already shown, the multiplier input binary point is between bits 27 and 28 and the output product binary point is between bits 28 and 29. This one bit shift to the left about the binary point is equivalent to multiplying the input multiplier by one half (i.e. the value of the coefficient). Thus, all of the coefficients whose values are one half (i.e MC1-MC5 and MC11-MC13) can employ the trivial multiplier cell to obtain the final product out of the pipeline at the appropriate time.

The advantages gained by using the trivial multipliers stem from the fact that these cells require less silicon area to implement and fewer control signals to operate. These include:

a. reduction in total silicon area required to implement real and imaginary multiplier pipeline.

b. reductions in control signal and clock line loads.

c. reductions in power consumption and heat dissipation.

3.8 Additional Encoding Schemes Investigated

As pointed out by Lyon's [7:422] it is possible to encode binary numbers in a manner which maximizes the number of zero bits. The conversion method used to do this is called the Canonical Signed Digit Code [15:164]. This code guarantees there will be at least one logical zero between every bit of unit magnitude. The advantage of using such a coding technique is that it requires fewer adders to do multiplication.

In an effort to maximize the number of zero modules contained in the WFT multiplier pipeline, a means of encoding the multiplicand dual-bit slice values was investigated. The results of this investigation revealed that the dual-bit values can be encoded to achieve the above objective. The resulting mathematical identity for this encoding process is given by:

$$-2^{2(n+1)} + \sum_{i=k+1}^{n} 2^{2i} + (2 \times 2^{2k}) = -2 \sum_{i=k}^{n} 2^{2i} \qquad (9)$$

where k=0,2,4,6,8,....n

Table 3 provides several examples of encoded dual-bit

multiplicands. Notice that, except for the first two examples, all of the encoded multiplicands contain a series of plus or minus twos. Either of these coded series would cause overflows in the WPT multiplier pipeline. Therefore, the recoding technique did not prove useful for this particular architecture. But, it could be used successfully in some parallel multiplier schemes or serial-multiplier architecture where the full range product is produced.

Original/ Encoded Dual-Bits	2ni	2(n-1)....24i	23i	22i	2i
O		-1	+2		
E		0	-2		
0		+1	-2		
E		0	+2		
0		-1	+1	+1	+2
E		0	+2	+2	+2
O		+1	-1	-1	-2
E		0	-2	-2	-2
O	-1	+1 ..... +1	+1	+1	+2
E	0	+2 ..... +2	+2	+2	+2
O	+1	-1 ..... -1	-1	-1	-2
O	0	+2 ..... +2	+2	+2	+2

Table 3.8 Additional Dual-Bit Encoding

IV. Pre- and Post-Addition Stages of WFT Processor

4.0 Overview

Addition and subtraction are the basic mathematical functions of both the pre- and post-addition stages. This chapter begins with an examination of the circuit selected to perform these functions. Next, a discussion of the resulting ADDSUB macrocell is provided. This cell includes additional hardware to buffer and control the inputs to the adder-subtractor circuit. And last, detailed analyses of the pre- and post-addition stages for the 16-point WFT are presented. These analyses demonstrate the operation and layout of both stages and serve as representative examples for the 15 and 17-point WFTs.

4.1 ADD/SUB Circuit

The adder-subtractor circuit was designed with an effort made to reduce not only the number of devices, but also the total parasitic capacitance associated with those devices [6]. The approach used was to reduce the number of intermediate boolean functions required to produce each output. Figure 4.1 summarizes the results of this effort. From the Karunaugh maps and subsequent boolean equations, it is seen that every output includes the XOR and XNOR functions of the two inputs, A and B. The reader may note that the manner in which the ones are grouped in the K-maps does not follow standard textbook procedures. However,

A	B	C_i	SUM	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A	B	B_i	DIFF	B_o
0	0	0	0	0
0	0	-1	1	-2
0	-1	0	1	-2
0	-1	-1	0	-2
1	0	0	1	0
1	0	-1	0	0
1	-1	0	0	0
1	-1	-1	1	-2

SUM BC_i

A \ BC_i	0 0	0 1	1 1	1 0
0	0	1	0	1
1	1	0	1	0

$SUM = \overline{C_i}\,(A \oplus B) + C_i\,\overline{(A \oplus B)}$
$= C_i \oplus (A \oplus B)$

DIFF BB_i

A \ BB_i	0 0	0 1	1 1	1 0
0	0	1	0	1
1	1	0	1	0

$DIFF = \overline{B_i}\,(A \oplus B) + B_i\,\overline{(A \oplus B)}$
$= B_i \oplus (A \oplus B)$

C_o BC_i

A \ BC_i	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	1	1	1

$C_o = C_i\,(A \oplus B) + A\,\overline{(A \oplus B)}$

B_o BB_i

A \ BB_i	0 0	0 1	1 1	1 0
0	0	1	1	1
1	0	0	1	0

$B_o = B_i\,\overline{(A \oplus B)} + \overline{A}\,(A \oplus B)$

Figure 4.1 ADD/SUB Truth Tables and Boolean Equations

the equations obtained completely define each output as summarized in the truth tables.

Transmission gate logic is used to implement the boolean equations in hardware, see Figure 4.2. First, the XOR and XNOR functions are formed from the inputs and then sent to the sum, difference, carry-out (CO), and borrow-out (BO) circuits. The final stage of both the sum and difference circuits are XOR functions and the carry-out and

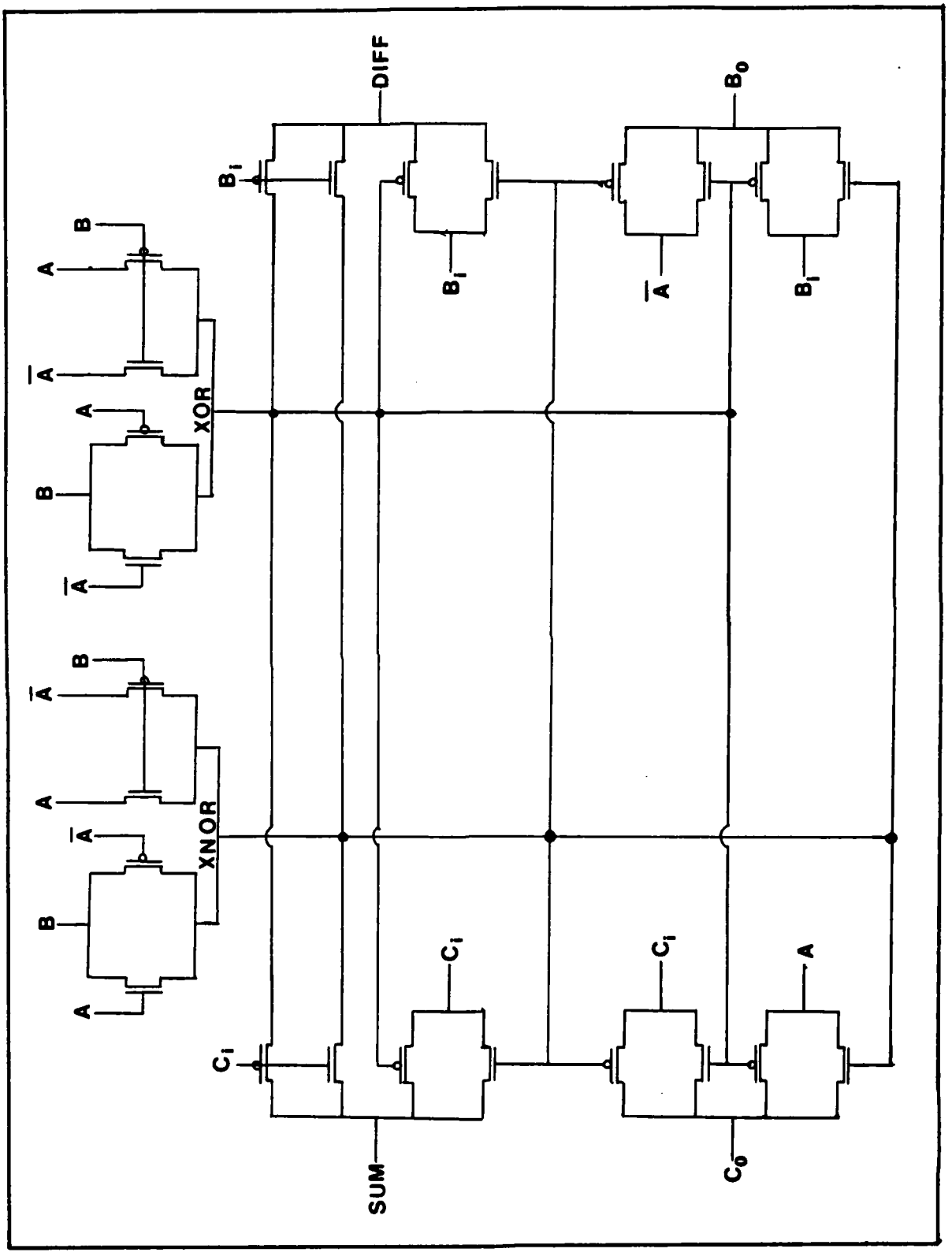


Figure 4.2. ADD/SUB CKT Boolean Equations Hardware

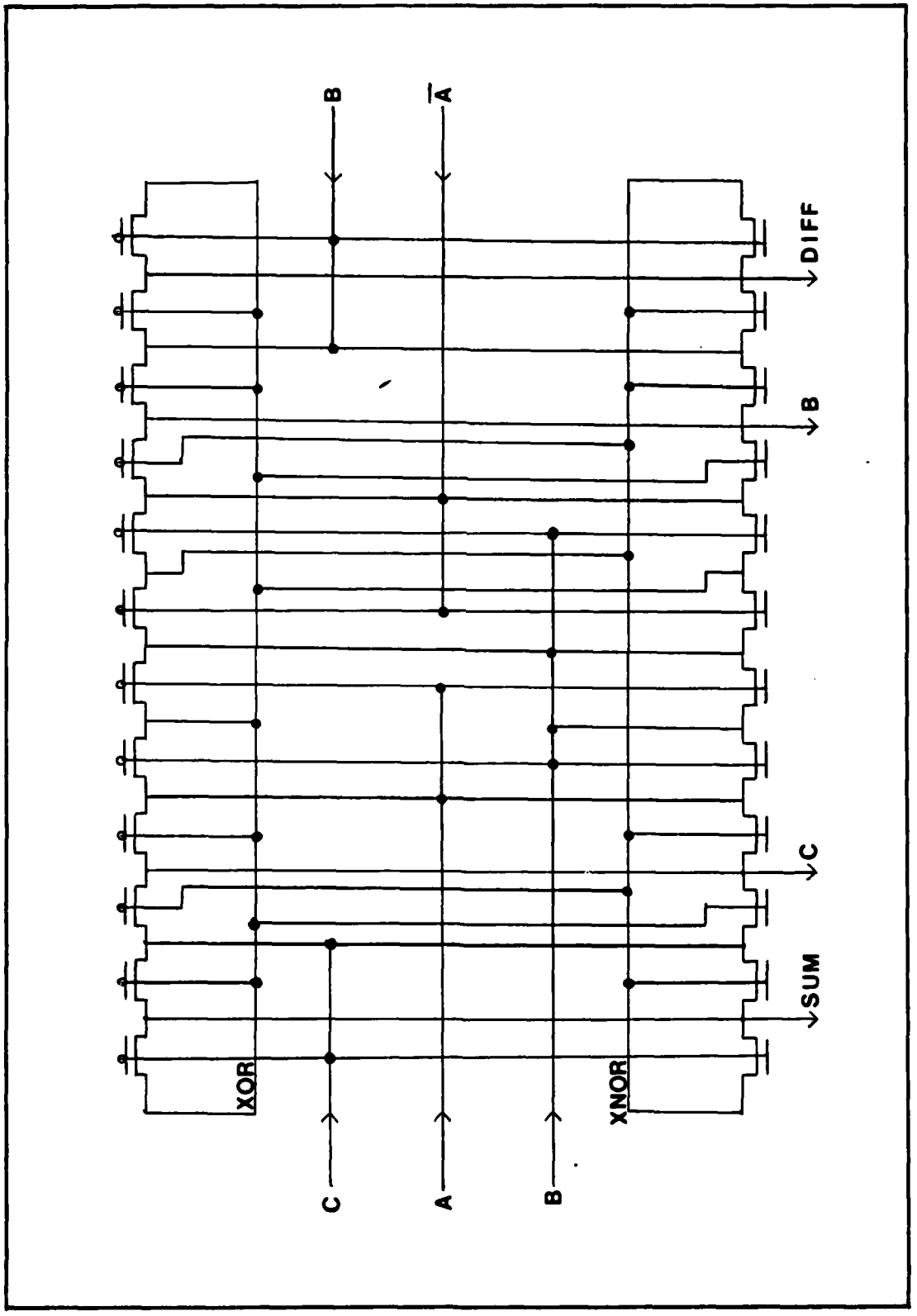


Figure 4.3. Optimized Layout of ADD/SUB CKT

borrow-out circuits are AND/OR functions constructed with two transmission gates. The total number of devices required is only 24. Another major advantage of the design is that the entire circuit can be laid out using only two strips of diffusion, see Figure 4.3. Therefore, the sources and drains of all the devices are shared. This results in a significant decrease in the parasitic capacitance of each device, which further enhances the circuit's performance. To attain high speed, the capacitance on the SUM and CYO must be small. Also, the number of devices can be reduced by 2 if the XNOR function is form with an inverter, but this would reduce the circuit's speed.

4.2 ADDSUB Macrocell

The ADDSUB macrocell is the primary cell used in both the pre- and post-addition stages. As depicted in Figure 4.4, it is composed of several registers and the adder-subtractor circuit discussed above. The cell's functional operation is as follows:

a. The two resettable MSFFs are used to store and control the borrow and carry bits. The reset signal is pulsed high on the rise of phi2 to reset the C_i and B_i bits to zero before the LSB of each serial data word is latched into the phi1 buffers.

b. On the rise of phi1 - C_i, B_i, B, A, and A bar, are input to the adder-subtractor circuit.

c. After the rise of phi1 and before the rise of the

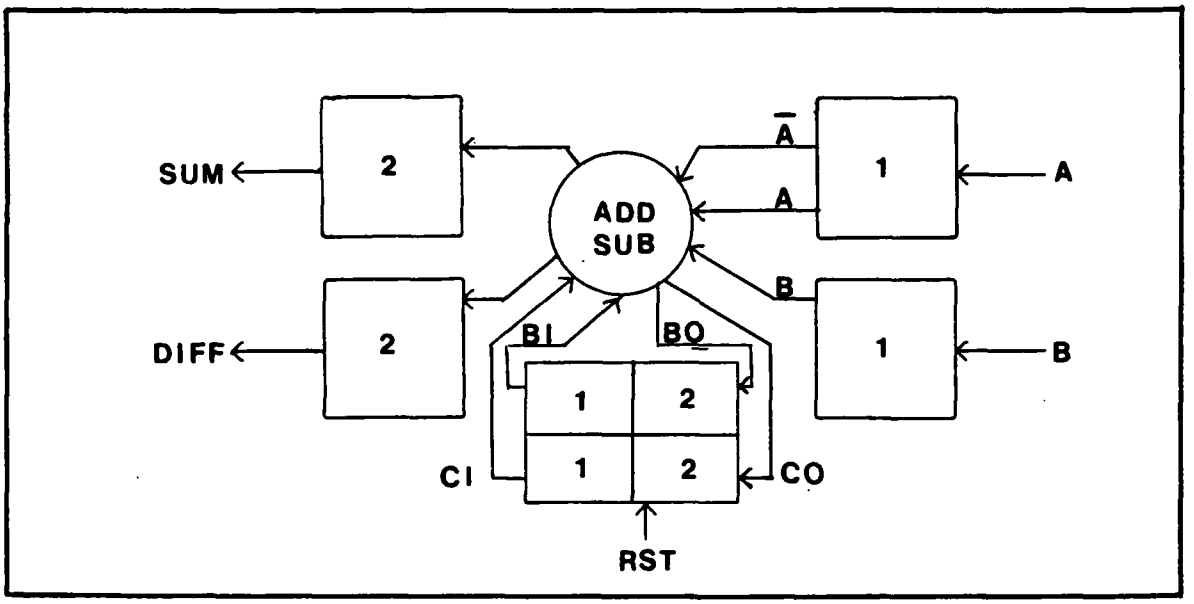


Figure 4.4 ADDSUB Macrocell

next phi2 clock - the adder-subtractor circuit computes the sum, difference, borrow-out, and carry-out.

e. On the rise of phi2 - the sum and difference are latched into the phi2 output buffers, and the carry and borrow-out results are latched into the phi2 portion of the MSFF.

f. On the next phi1 clock - the above process is repeated for as many clock cycles as necessary to serially compute the sum and difference of A and B.

The phi1 input buffers and MSFFs are designed to stage up the inputs to the adder-subtractor circuits.

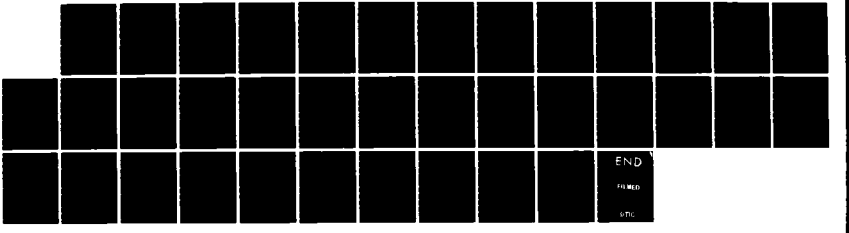
4.3 Pre-Addition Stage of 16-Point WFT

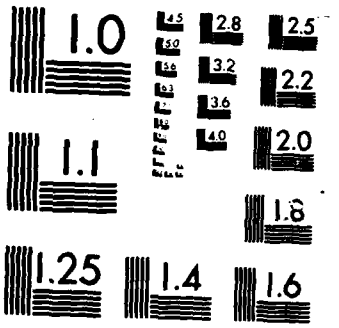
The pre-addition stage is the first stage in the calculation portion of the WFT pipeline. Its objective is to compute the results of the matrix multiplications CV_i, where C is the pre-addition matrix and V_i is the single column matrix of input data points. Since all of the elements of the C matrix are +1 or -1, the product of CV_i can be attained by doing successive addition and/or subtraction operations. The maximum number of additions/subtractions required to compute a given elemental product (i.e. individual products obtained from multiplying V_i by each row of the "C" matrix) is a function of the number of unit vectors contained in its respective row of the C matrix. This is an important property in the WFT "pipelined" architecture because it dictates the number of addition/subtraction levels that must be implemented to compute and synchronously propagate each elemental product through the pre-addition pipeline.

The required additions and subtractions in the pre-addition stage, for both the real and imaginary input data point values of the 16-point WFT, are given in table 4.1. Using a straight forward approach, four levels of addition/subtraction would be necessary. However, further examination of the mathematical operations performed on the results attained in level four reveals that they are multiplied by coefficients of the same magnitude and sign in the multiplication stage. Therefore, level four in the

AD-A163 943 ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. P W COUTEE 2/2

UNCLASSIFIED DEC 85 AFIT/GE/ENG/85D-11 F/G 9/5 NL





Level One	Level Two	Level Three	Level four
R100 = V0 + V8	R200 = R100 + R108	R300 = R200 + R202	R400 = R300 + R302
R101 = V0 - V8	R201 = R100 - R108	R301 = R200 - R202	R401 = R300 - R302
R102 = V1 + V9	R202 = R112 + R104	R302 = R206 + R204	
R103 = V1 - V9	R203 = R112 - R104	R303 = R206 - R204	
R104 = V2 + V10	R204 = R102 + R110	R304 = R205 + R207	
R105 = V2 - V10	R205 = R102 - R110	R305 = R205 - R207	
R106 = V3 + V11	R206 = R106 + R114	R306 = R209 + R211	
R107 = V3 - V11	R207 = R106 - R114	R307 = R208 + R210	
R108 = V4 + V12	R208 = R103 + R115		
R109 = V4 - V12	R209 = R103 - R115		
R110 = V5 + V13	R210 = R111 + R107		
R111 = V5 - V13	R211 = R111 - R107		
R112 = V6 + V14	R212 = R105 + R113		
R113 = V6 - V14	R213 = R105 - R113		
R114 = V7 + V15			
R115 = V7 - V15			

Table 4.1 Pre-Addition Arithmetic for 16-Point WFT

pre-addition was deleted by replacing the R400 and R401 inputs into the multiplier stage with R300 and R302, respectively. Then, to attain the transformed data points, the sum and difference of their respective products, output from the multiplier stage, are added in the post-addition stage.

The real half of the resulting three level pre-addition stage is shown in Figure 4.5. There are no hardware differences between the imaginary and real portions of the pre-addition stage. The only difference is that one's inputs are the imaginary magnitudes of the complex data point and the other's inputs are the real magnitudes. Each level is composed of ADDSUB cells and MSFFs which are used to maintain synchronization between data points as they are piped through the pre-addition stage. Reset control of the

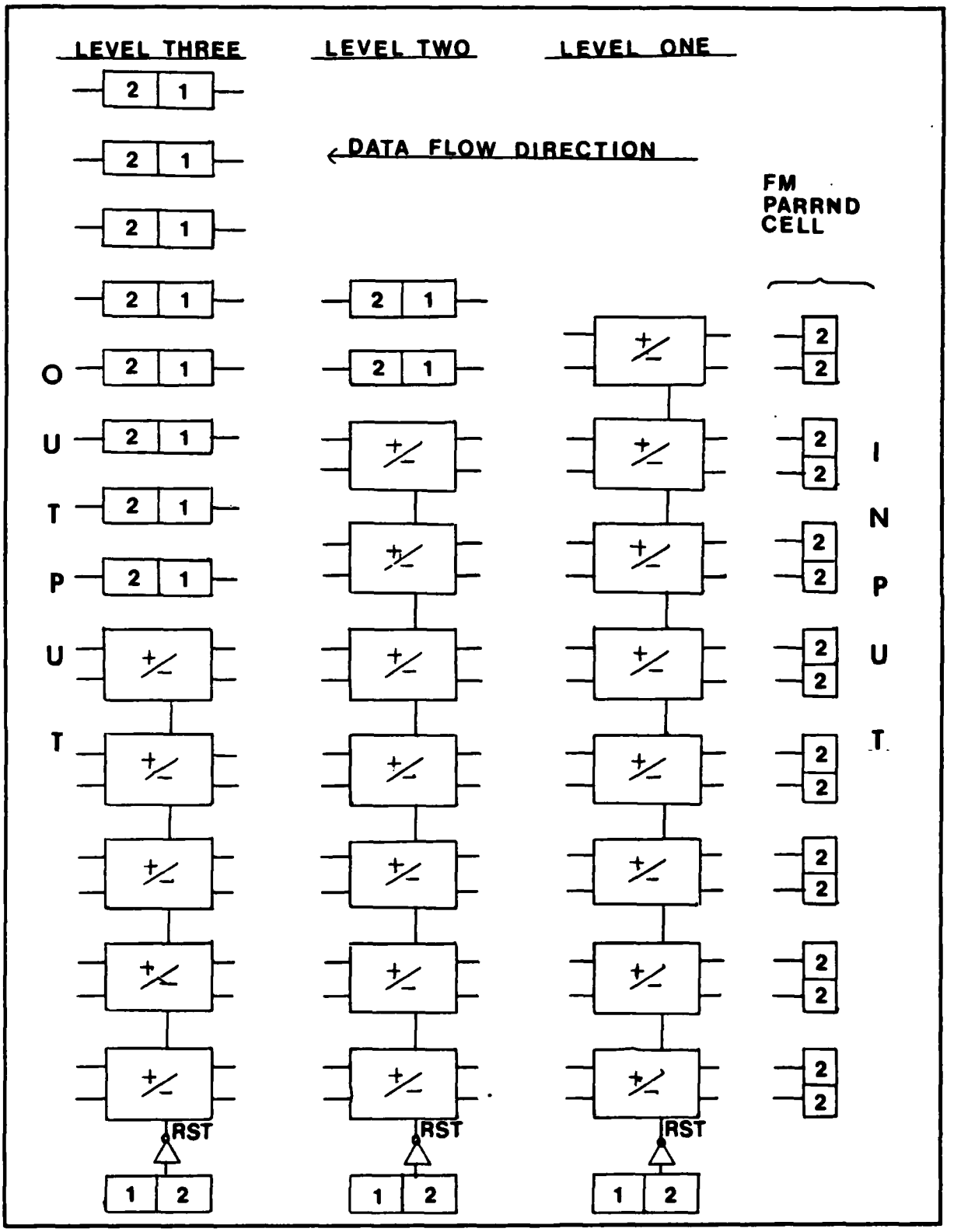


Figure 4.5. 16-Point WFT Pre-Addition Stage Configuration

adders is provided by the MSFFs at the bottom of each level. The input control word to the initial FF is the same bit length (i.e. 32 bits) as the data word input to the pre-addition stage. It consists of one zero in its LSB position followed by 31 ones. Timing is such that the zero bit input to the first FF coincides with the arrival of the data words LSBs at the phi2 registers in the PARRND cells (see Chapter 5). Propagation of the zero through each level assures the ADDSUB cell's carry and borrow bits are reset to zero before calculating the sum and difference of the next set of data words in the pipeline.

Recall that the input to the ADDSUB cells occurs on phi 1 and the results are output on phi2. This is the opposite of what occurs in the multiplier pipeline where data is input on phi2 and output phil. Therefore, to interface the pre-addition stage with the multiplication stage, an extra phil register is appended to the end of each ADDSUB and MSFF in the last level of the pre-addition stage.

Also notice that the control word used is identical to that required for the multiplication stage. All that need be done to use the same control word for both stages is to ensure it arrives at the multiplication stage in the proper time frame. This is accomplished by removing the phi2 latch at the multiplier input of each multiplier cell in the least significant position of the multiplier array.

Eliminating the fourth level of addition/subtraction voided the requirement for inclusion of an extra column of

hardware in the pre-addition stage. The advantages gained in doing so include: (1) a 25 percent reduction in silicon area, (2) a one cycle decrease in the pipeline latency, (3) a reduction in total power consumption and (4) one less sign extension on the inputs which translate to an extra bit of precision in the results.

4.4 Post-Addition Stage of 16-Point WFT

The post-addition stage is the last arithmetic stage of the WFT process. The circuit computes the final transformed outputs which are obtained from the matrix product of AP, where A is the post-addition matrix and P is the single column product matrix output from the multiplication stage. This stage is similar to the pre-addition stage in that all of the elemental values in the A matrix are unit vectors. Thus, the above matrix product can be attained through successive levels of addition. However, unlike the pre-addition matrix, C, which only contains the real unit vectors, the post-addition matrix contains both real and imaginary unit vectors (i.e. +1, -1, +j, and -j). Also, as shown in table 4.2, the last level of additions and subtractions in the post-addition stage requires the intermixing of results from the real and imaginary hardware to produce the transformed data points (V_oX = VRX + VIX). Therefore, the real and imaginary hardware of the post-addition stage is not completely separable. This is the only point where the imaginary and real hardware merges.

Level One	Level Two	Level Three
R100 = RP03 + RP05	R200 = R104 + R106	VR1 = R200 + I204
R101 = RP03 - RP05	R201 = R104 - R106	VR15 = R200 - I204
R102 = RP13 + RP11	R202 = R105 + R107	VR2 = R100 + I102
R103 = RP13 - RP11	R203 = R105 - R107	VR14 = R100 - I102
R104 = RP04 + RP06	R204 = R108 + R110	VR3 = R203 - I207
R105 = RP04 - RP06	R205 = R108 - R110	VR13 = R203 + I207
R106 = RP08 + RP07	R206 = R109 + R111	VR4 = RP02 + IP10
R107 = RP09 - RP07	R207 = R109 - R111	VR12 = RP02 - IP10
R108 = RP12 + RP14		VR5 = R202 + I206
R109 = RP12 - RP14		VR11 = R202 - I206
R110 = RP15 + RP16		VR6 = R101 + I103
R111 = RP15 - RP17		VR10 = R101 - I103
		VR7 = R201 - I205
		VR9 = R201 + I205
		VR0 = RP00 + RP01
		VR8 = RP00 - RP01
I101 = IP03 - IP05	I201 = I104 - I106	VI15 = I200 + R204
I102 = IP13 + IP11	I202 = I105 + I107	VI2 = I100 - R102
I103 = IP13 - IP11	I203 = I105 - I107	VI14 = I100 + R102
I104 = IP04 + IP06	I204 = I108 + I110	VI3 = I203 + R207
I105 = IP04 - IP06	I205 = I108 - I110	VI13 = I203 - R207
I106 = IP08 + IP07	I206 = I109 + I111	VI4 = IP02 - RP10
I107 = IP09 - IP07	I207 = I109 - I111	VI12 = IP02 + RP10
I108 = IP12 + IP14		VI5 = I202 - R206
I109 = IP12 - IP14		VI11 = I202 + R206
I110 = IP15 + IP16		VI6 = I101 - R103
I111 = IP15 - IP17		VI10 = I101 + R103
		VI7 = I201 + R205
		VI9 = I201 - R205
		VI0 = IP00 - IP01
		VI8 = IP00 + IP01

Table 4.2 Post-Addition Arithmetic for 16-Point WFT

The hardware floor plan of the post-addition stage is shown in Figure 4.6. Its function is similar to the pre-addition stage.

4.5 Macrocell Size and Density

The ADDSUB macrocell is 207 x 211.5 lambda and its density in a 3 micron technology is approximately 91,942 devices/cm^2.

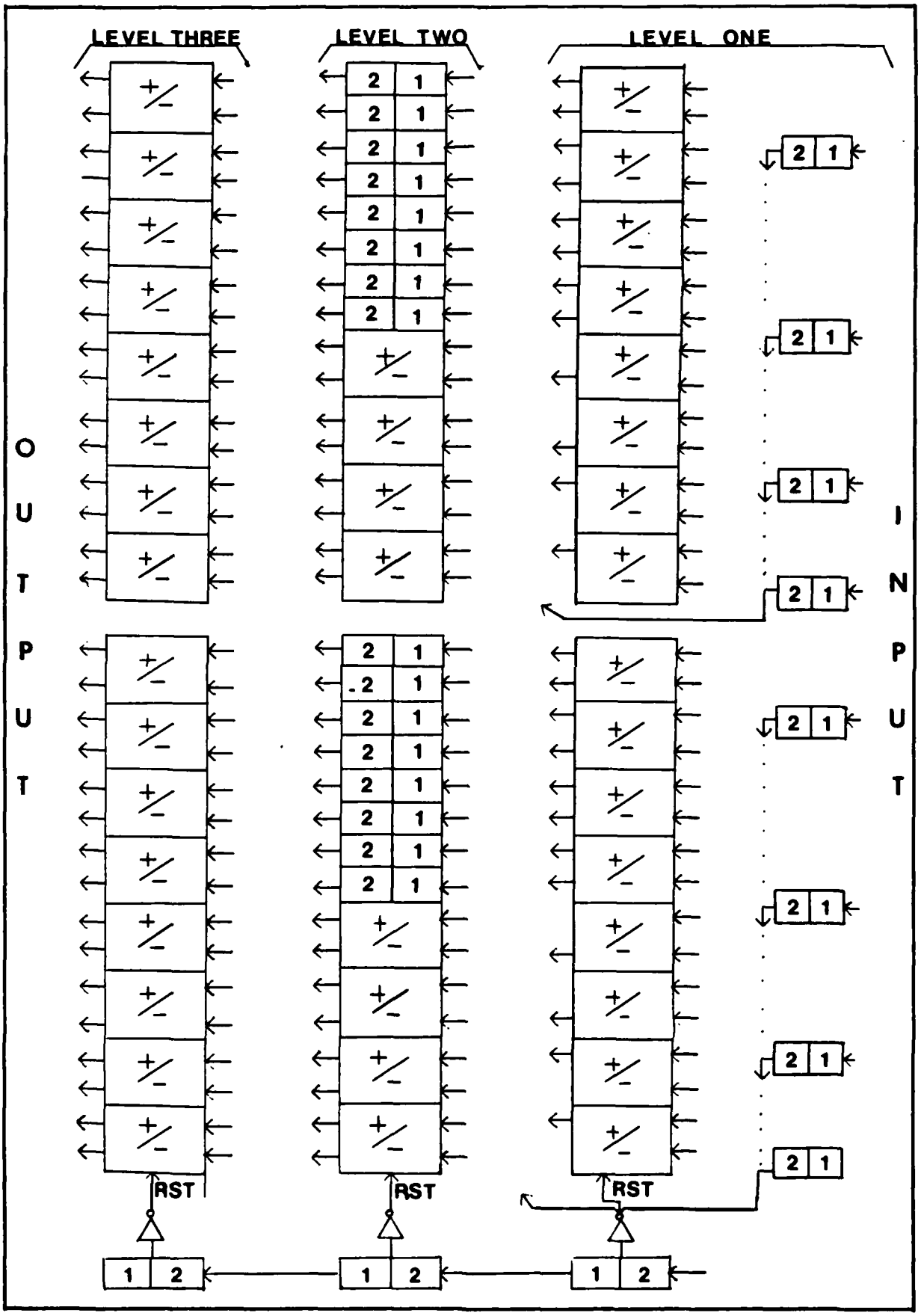


Figure 4.6. 16-Point WFT Post-Addition Stage Configuration

V. Input/Output Stage of 16-Point WFT Processor

5.0 Overview

The primary function of the input and output (I/O) stages is to provide a buffered I/O that is capable of continuously transferring blocks of data words into and out of the WFT arithmetic processor pipeline. These stages also perform parity generation/checking, rounding, generation of scale factor codes, and data word formatting. The following discussion is specific to the 16-point transform, but the only differences between it and other size transforms are the array sizes and some timing details. This chapter is divided into two sections which discuss the macrocells and layout of the input and output stages, respectively.

5.1 Input Stage

The input stage of the processor pipeline performs four functions. The first is to provide a means of accepting each set of 16 data words from the input pads, one word at a time in a bit-parallel fashion, and then simultaneously latch all 16 words into the serial data path for processing. The unit macrocell used to accomplish this function is a latchable Parallel-In Serial-Out bit data register called PISO.

The next two functions of the input stage are to check the parity of the input data word and extend the data word

length by adding zero fills and sign extensions. Both of these functions are conducted in parallel by a macrocell called PARZER. The last function is to generate a parity error signal if any one of the 4080 data words received fails to have odd parity. The macrocell used to generate the parity error signal is called PARERR.

5.1.1 PISO Cell. The PISO cell logic diagram is shown in Figure 5.1. As illustrated, this cell is composed of two master slave flip-flops and three transmission gates. Access to, and control of the data path flow are provided by nine terminals defined as:

a. Two input terminals, parallel-in (PI) and serial-in (SI).

b. Two output terminals, parallel-out (PO) and serial-out (SO).

c. Three control signal terminals, shift down (SDP), shift right (SRP), and latch (LP).

d. Two clock terminals, phi1 and phi2.

Each cell can hold two bits of data, one in MSFF1, the parallel data path, and the other in MSFF2, the serial data path.

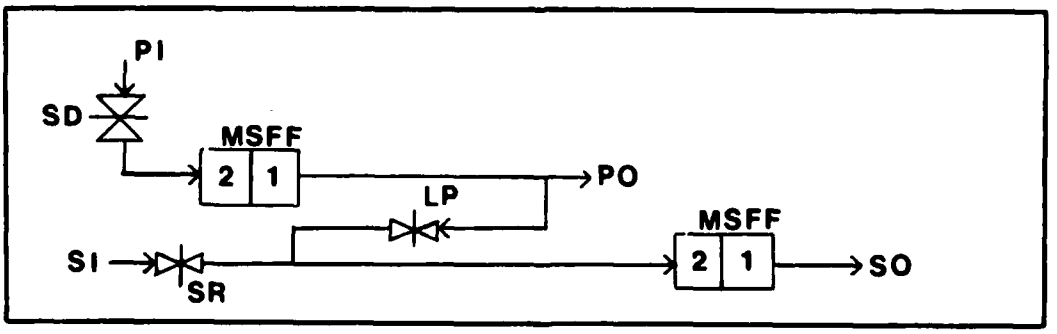


Figure 5.1 Logic Diagram of SIPO Cell

5.1.1.1 PISO Array. The complex input data is divided into two words, real and imaginary, of 24 bits. Each format is depicted in Figure 5.2. Therefore, two 16 by 24 arrays of the PISO cell are needed to facilitate the parallel loading and serial input of the 16 data words into the processor's pipeline. In reference to Figure 5.3, the parallel inputs of the top row are connected to the input data pads, and the parallel inputs of all other rows are connected to the parallel outputs of the row above them. The serial outputs of the last column are connected to the inputs of their respective PARZER cell, and the serial outputs, columns 1 through 15, are connected to the serial inputs of rows 2 through 16 respectively. Note that the serial inputs of column one and the parallel outputs of row 16 are not connected and have no effect on the functional operation of the array. Although not shown in the figure, all LP terminals are connected to a common control line as are phi1 and phi2. As drawn, the data flow direction is from top to bottom and left to right. Data flow is controlled by the SDP, LP, and SRP signals whose functions are as follows:

a. SDP is a half clock frequency signal. It is high during odd clock cycles, beginning with cycle one, and is low during even clock cycles. At the beginning of clock cycle 32, all 16 complex data words are loaded into the

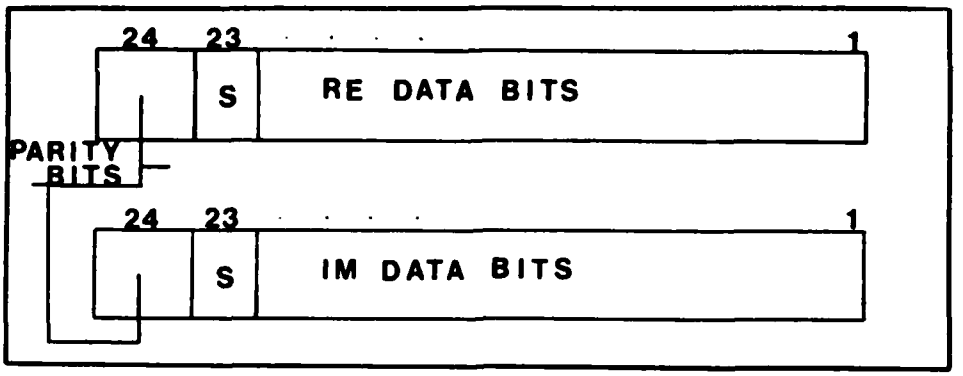


Figure 5.2 Input Data Word Formats

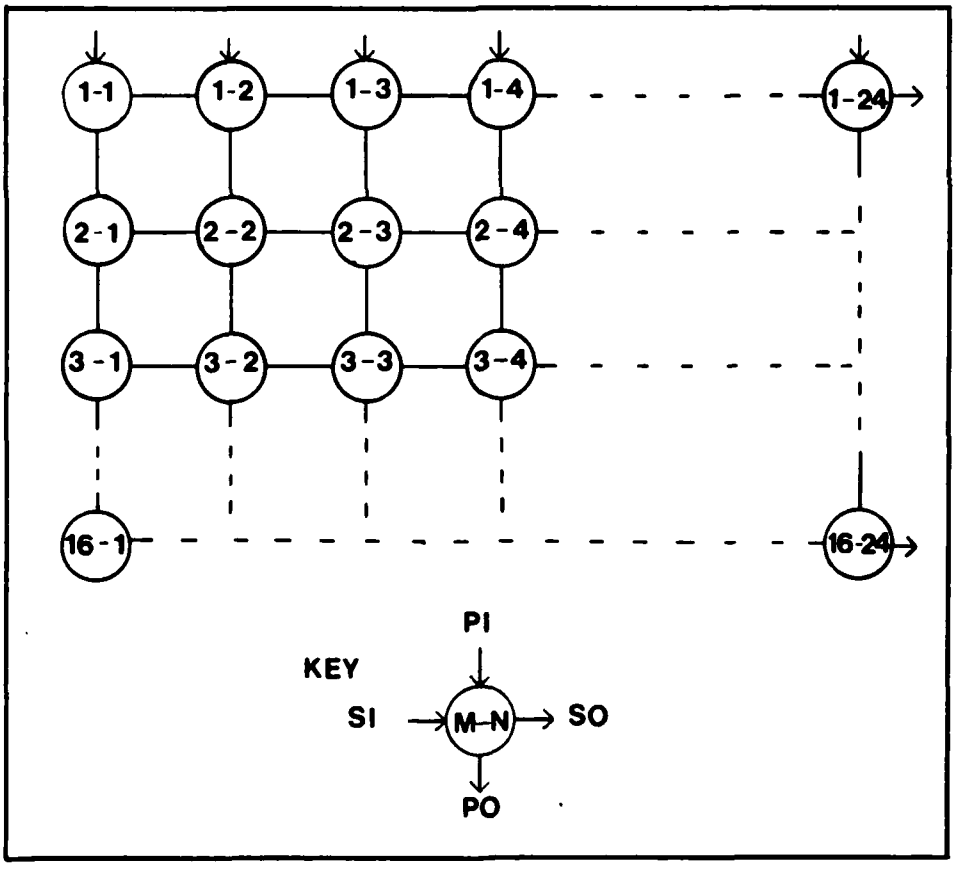


Figure 5.3 PISO Array

SMSFF1s of the two PISO arrays, real and imaginary. (NOTE: It is assumed that a new valid data word is stable at the input pads during each odd cycle).

b. LP is a pulse signal which is active high during clock cycle 32. The purpose of this signal is to latch the data in the parallel data registers, SMSFF1s, into the serial data register, SMSFF2s.

c. SRP is a level signal that is set high and reset low as a function of the input scale factor received from the previous processor in the overall 4080-point WFT pipeline, or from the external host system. When SRPISO is set, the data in its serial registers are clocked into their respective PARZER cells.

5.1.2 PARZER and PARERR Cell. The logic diagram of the PARZER cell is depicted in Figure 5.4. This cell is divided into two functional sections. The function of the top half of the cell is to perform a parity check on the 24-bit data word being transmitted in from the PISO cell. The output of this circuit is connected to one of the 16 inputs of the PARERR cell, which generates an error signal if any of the data words have even parity. The bottom half of the PARZER cell provides the necessary circuitry to change the data word length from 24 to 32 bits by adding five zeroes to the least significant positions and four sign extensions of the most significant bit.

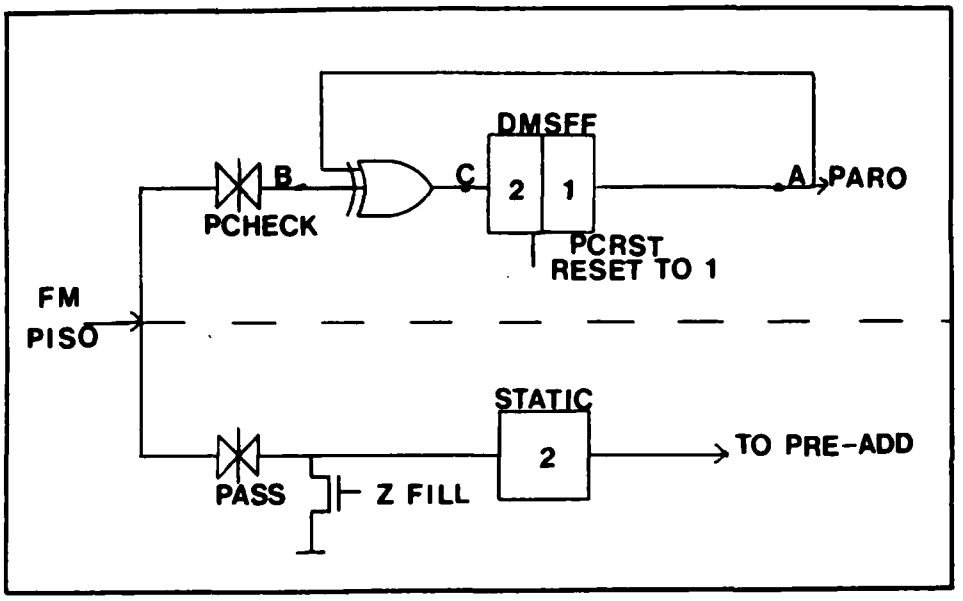
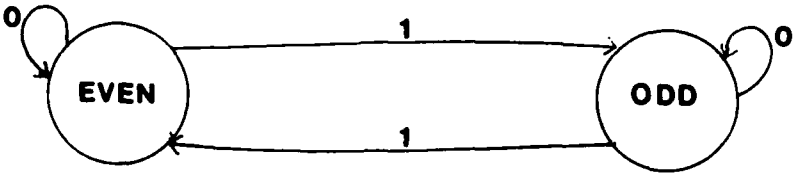


Figure 5.4 Logic Diagram of PARZER Cell

5.1.2.1 Parity Checker Circuit. The parity checker is a sequential finite state circuit. The state of the circuit is stored in the DMSFF which is initialized to an even parity state (A = 1 yields even parity and A = 0 yields odd parity) by the parity check reset pulse signal, PCRST, just prior to the beginning each data word check. As illustrated by the state diagram and truth table in Figure 5.5:

a. If the present state is even and; (1) the input is a zero, then the next state is even; (2) the input is a one, then the next state is odd.

b. If the present state is odd and; (1) the input is a zero, then the next state is odd; (2) the input is a one, then the next state is even.



PRESENT STATE A	INPUT BIT B	NEXT STATE A+1
1	0	1 (EVEN)
1	1	0 (ODD)
0	0	0 (ODD)
0	1	1 (EVEN)

Figure 5.5 Parity Checker (a) state diagram, (b) truth table.

The truth table also shows that the next state is an exclusive OR function of the present state and next input bit. Data flow into the parity checker passes through the transmission gate which is controlled by the parity check signal, PCHECK. This is a level signal which goes high at the same time as SRPISO but is one clock cycle longer, 24 total. This permits all 24 bits of the data word, including the parity bit, to pass through the parity check circuit. As with SRP, PCHECK timing is a function of the input scale factor.

5.1.2.2 Parity Error Circuit. The parity error circuit is designed such that the parity error bit is set any time a parity error is detected in any of the the 4080-point transform data words. It does not indicate the specific word or block of 16 words where the error

occurred. As shown in Figure 5.6, each of the parity output bits, PAR01 through PAR016, from the PARZER cell, are connected to devices A1 through A16 in the PARERR cell.

The functional operation of the cell is controlled by two pulse signals, PALATCH (parity latch) and PERRST (parity error reset). PALATCH is a normally low signal that is pulsed high for one cycle just after the 24th bit of each data word is check by the parity check circuit. During this cycle, if one or more parity error is detected (i.e. any PAROXX = 1) in the parity check circuit, the gate of device D will be pulled low, thus setting the parity error bit high. When the PLATCH signal is low, it effectively buffers the PARERR cell from the intermediate states of the parity out signals, PAROXX, by holding devices A1 through A16 in an open state, and device C in a closed state. PERRST is also a normally low signal that goes high for one cycle just prior to the beginning of each 4080-point transform. This causes the parity error bit to be reset to zero. Both the real and imaginary sections each contain one PARERR cell. Their outputs are sent to an OR gate whose output is connected to the parity out pad.

5.1.2.3 Word Adjust Circuit. To increase the accuracy of the transformed points, and to insure that an overflow does not occur during the transform calculation process, the 24-bit input data word is adjusted to 32 bits. This adjustment is accomplished by adding zeroes to the least significant positions and sign extension in the

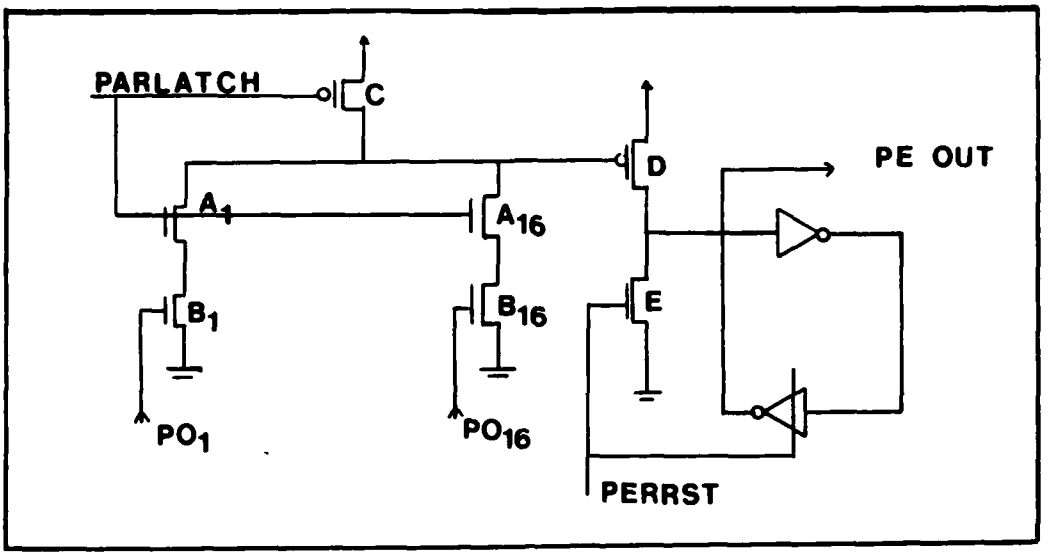


Figure 5.6 Logic Diagram of PARERR Cell

most significant positions. The precise number of zero fills and sign extensions is a function of the scale factor associated with each 4080-point transform. This scale factor identifies the minimum number of sign extensions already contained in the input data words and is provided by the previous stage of the overall 4080-point transform pipeline or the host system. This information, coupled with the fact that there must be at least four sign extensions in each 32-bit word (to prevent overflow conditions from occurring during the calculation process), sets the number of zeroes and sign extensions to be added to all of the 4080 input words (see table 5.1).

The circuit used to modify the input data words is depicted in Figure 5.4. As shown, it contains only one transmission gate, an n-device, and a static phi2 storage register. There are two control signals that determine the output from the phi2 register, zero fill (ZFILL) and PASS.

Scale Factor	# Zero Fills	# Sign Extensions
0	5	4
1	6	3
2	7	2
3	8	1
4	9	0
5	9	0
6	9	0
7	9	0

Table 5.1 Scale Factor Function

To add zeroes in the least significant bit positions, the zero fill signal is set high from 4 to 9 cycles at the beginning of each data word transfer into the calculation pipeline. As ZFILL is reset, the PASS signal is set high for 23 cycles in coincident with SRPISO. This causes the 23 input data bits, including the sign, to be transferred into the pipeline behind the zero fill bits. When the PASS signal is reset, sign extensions will continue to be passed into the pipeline (note: the sign bit, 23, is stored in the static phi2 register) until the ZFILL is again set for the beginning of the next 32-bit data word to be input to the pipeline. As with SRPISO and PCHECK, both the PASS and ZFILL signals are functions of the scale factor.

5.2 Output Stage

The output stage of the processor encompasses four unit macrocells (PARRND, SIPO, XORSCAL, and OUTDRIV) to accomplish five functions. The first two functions, parity bit generation and rounding, are accomplished by the PARRND cell, whose output is connected to the serial input of the PISO cell. The next function, which is to collect the serial data stream result and provide a buffered parallel output path, is accomplished by the SIPO cell. The last two functions, scale code generation and final output driver, are achieved by the XORSCAL and OUTDRIV cells.

5.2.1 PARRND Cell. The input of the PARRND cell is the 32-bit result out of the post-addition stage. Of the 32 bits, only the 23 most significant bits (i.e. bits 10-32) are kept. The purpose of the PARRND cell is to round at bit location 9 and to append a parity bit in location 24. As illustrated in Figure 5.7, this cell consists of two interacting circuits. The bottom half of the circuit performs the rounding function, the upper half performs the parity bit generation function, and the center section selects the output data path source.

5.2.1.1 Rounding Circuit. To perform the rounding operation, a 1 must be added to bit location 9, and the resultant carry propagated through and added to the input bit stream until the first zero is encountered. To demonstrate how this operation is implemented serially,

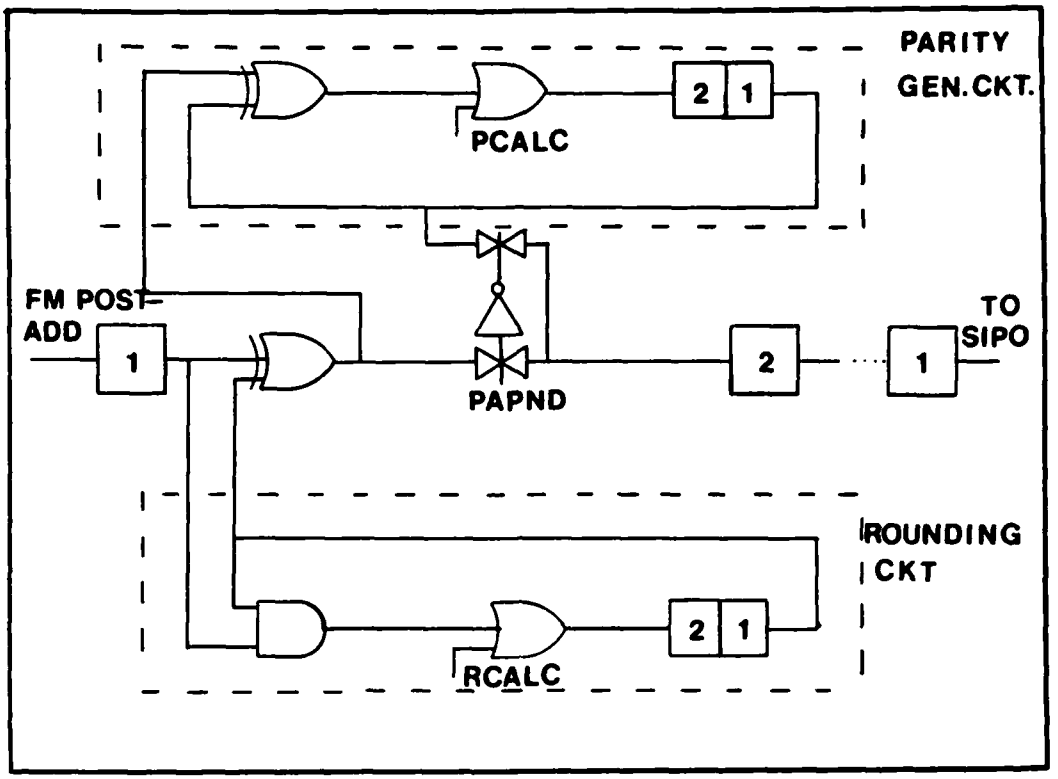


Figure 5.7 Logic Diagram of PARRND Cell

consider the example illustrated in Figure 5.8. As shown there, the next carry bit and the final result are the AND and XOR functions, respectively, of the current input and current carry bits.

The functional operation of the rounding circuit is as follows:

a. When RCALC bar is set high, the input to the carry bit data register will always be high, thus insuring that the initial carry or rounding bit is a one.

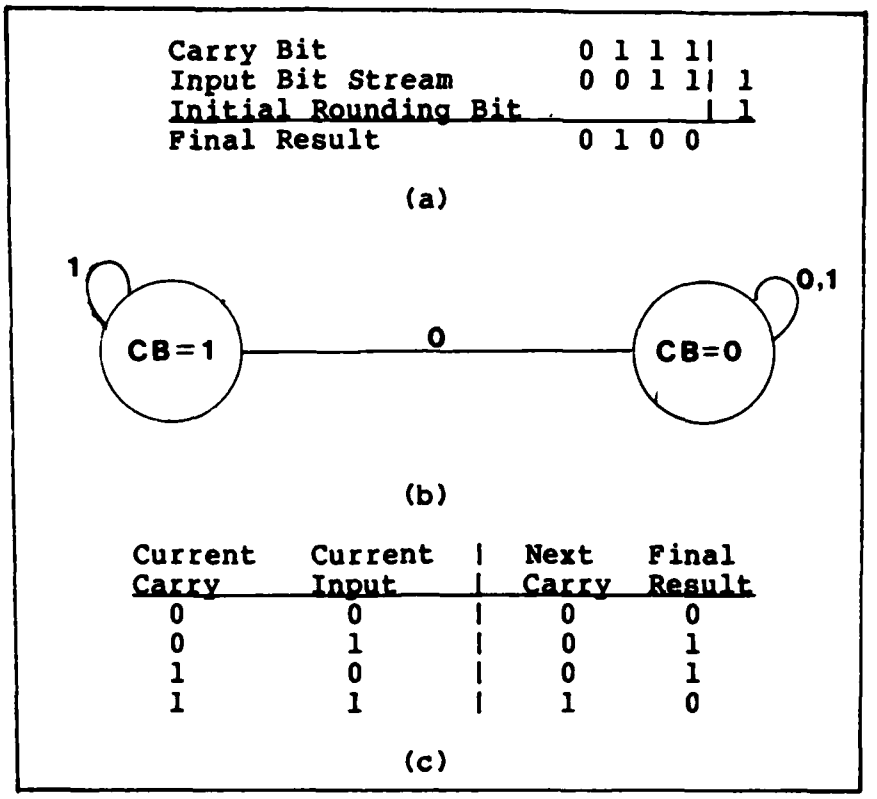


Figure 5.8 Rounding Circuit (a) sample operation, (b) state diagram, (c) truth table.

b. RCALC bar is reset low one cycle before bit #10 of the input data word arrives. Therefore, when bit #9 arrives at the input to the XOR gate, the LSB of the result [i.e. LSB = (rounding bit AND input bit #8) XOR bit #9] is formed at the output of the XOR gate.

c. To insure complete propagation of carry bits, RCALC bar remains low for 24 cycles. In general, the result out of the rounding circuit can be expressed logically as:

$$R_n = [\ 1 \text{ AND } I_{(m-n)} \text{ AND } I_{(m-n)+1} \cdots \text{AND } I_{(m-1)}] \text{ XOR } I_m \quad (10)$$

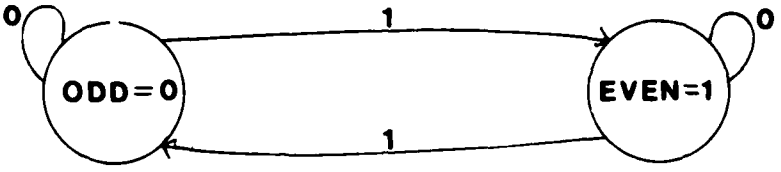
where:

R_n is defined as the result at bit location n and n = 0,1,2,3,.....,22.

I_m is defined as the input bit at location m and m = 9,10,11,12,...,31.

[Note, the n and m indexes are sequentially indexed simultaneously (e.g. when n=1, m=11.)]

5.2.1.2 Parity Generation Circuit. The parity generation circuit is a sequential finite state circuit that monitors the most significant 23 data bits output from the rounding circuit to determine whether the bits contain an even or odd number of ones. To insure the final output data word is odd parity, a one is appended if the circuit's final state is even, and a zero is appended if it is odd. The state diagram and truth for the parity generator are shown in Figure 5.9. To achieve odd parity generation, the initial state must be even. The truth table shows that the next output is the exclusive OR function of the present output and current input.



(a)

Current Input	Present State/Output	Next State/Output
0	Odd/0	Odd/0
0	Even/1	Even/1
1	Odd/0	Even/1
1	Even/1	Odd/0

(b)

Figure 5.9 Parity Bit Generation (a) state diagram, (b) truth table

The circuit's functional operation is as follows:

a. When PCALC bar is set high the input to the parity bit register is maintained at 1, thus insuring that the initial state is even.

b. PCALC bar is reset low one cycle after RCALC bar is reset low (note: this is when the first result bit, R_0, from the rounding circuit appears at the input to the XOR gate) and remains low for 23 clock cycles. At the end of the 23rd cycle, the parity bit to be appended to the data word will be in the storage register.

5.2.1.3 Output Data Path. As illustrated in Figure 5.7, the output data path is chosen by the 2:1 mux that is controlled by a pulse signal called parity append,

PAPND. This signal is pulsed high (rises and falls on phi1) for one cycle after the parity bit is generated, thereby appending the parity bit in the 23rd position of the output data word. Otherwise, the output from the rounding circuit is passed to the next cell, SIPO. However, as explained in the next section, only the last 24 data bits output from the PARRND cell are saved in the SIPO cell.

5.2.1.4 PARRND Array. Two single column arrays of 16-bit slices are used to simultaneously receive and process the 32 data words, 16 real and 16 imaginary, coming out of the post-addition stage.

5.2.2 SIPO Cell. A logic diagram of the SIPO cell is depicted in Figure 5.10. As should be expected, this cell is the dual of the PISO cell. In contrast to the PISO cell, the SIPO cell's serial input register, MSFF1, is latchable into its parallel output register, MSFF2. Access and control of the cell is provided by:

a. Two input terminals, serial-in (SI) and parallel-in (PI).
b. Two output terminals, serial-out (SO) and parallel-out (PO).
c. Three control signal terminals, shift down (SDS), shift right (SRS), and latch (LS).
d. Two clock terminals, phi1 and phi2.

Each cell stores two bits of data, one in MSFF1, the serial data register, and one in MSFF2, the parallel data register.

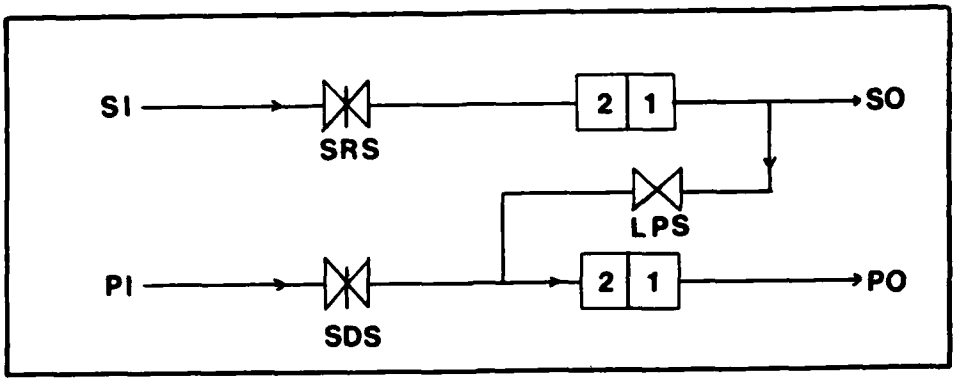


Figure 5.10 Logic Diagram of SIPO Cell

5.2.2.1 SIPO Array. Two 16 x 24 SIPO cell arrays are required to output both the real and imaginary parts of the transformed data points. As illustrated in Figure 5.11, the serial inputs of the first column are connected to the serial outputs of the PARRND array, and serial inputs of the remaining columns are connected to the serial outputs of the column to their left. Also, the parallel outputs of the most significant eight bits in the bottom row are connected to the inputs of XORSCAL cells, and the remaining outputs in that row are connected to output drivers. The parallel outputs of rows 1 through 15 are connected to the parallel inputs of the rows just below them. The serial outputs of column 16 and parallel inputs of row 1 are not connected and do not effect the operation of the array. Data flow is from left to right, then top to bottom, and is controlled by SRS, LS, and SDS. The set and reset sequences of these signals are as follows:

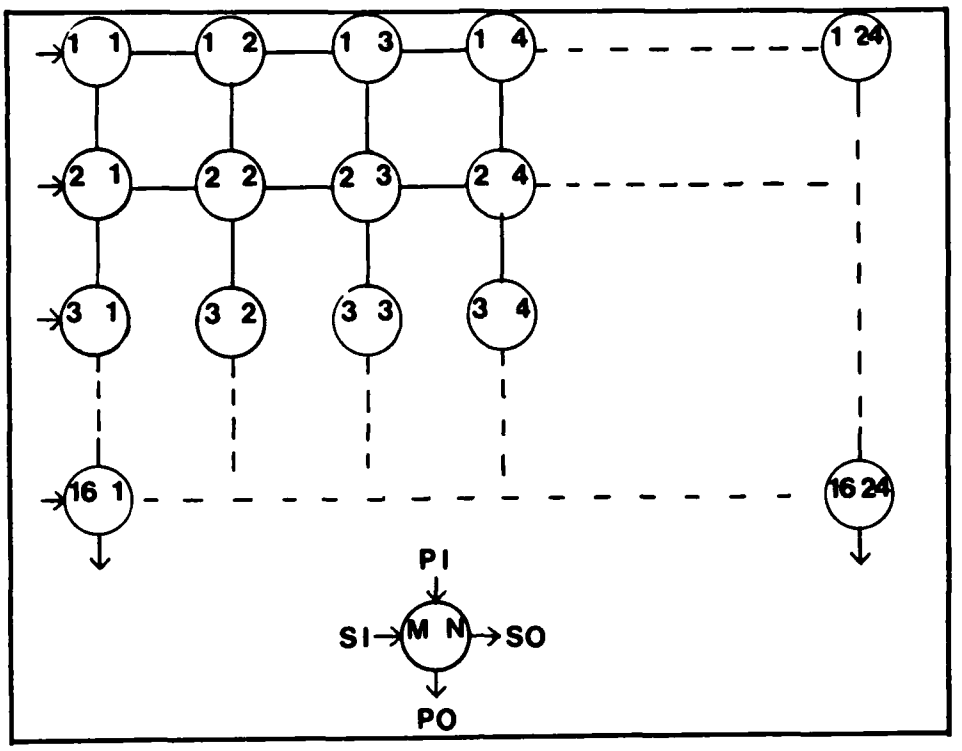


Figure 5.11 SIPO Array Configuration

a. SRS is a level signal that is set high in coincident with PCALC and remains set for 24 cycles. This enables the serial loading of the 23 most significant bits of the result and the parity bit from the PARRND cells into serial data bit registers of the SIPO cells.

b. LP is a pulse signal that is pulsed high for one cycle just after the parity bit is loaded into the serial data registers. This causes the data words in the serial data registers to be latched in the parallel data registers.

c. SDS is a continuous half frequency signal whose function is to output the transformed data words in a bit-parallel fashion. To insure proper operation of the array, SDS must be low when LS is high.

5.2.3 XORSCAL Cell. The primary objective of the XORSCAL cell is to compute the real and imaginary scale codes, RE_m and IM_m respectively, that are used by the scaling encoder circuit to calculate the 4080-point scale factor (i.e. the minimum number of sign extensions contained in any of the 4080 transformed data points). The algebraic boolean relations used to formulate these codes are as follows:

$$RE_m = (RRB_n \text{ XOR } RRB_{n-1})$$

$$IM_m = (IRB_n \text{ XOR } IRB_{n-1})$$

where:

$RRB_{n,n-1}$ is defined as the real result bits at bit locations n and n-1.

$IRB_{n,n-1}$ is defined as the imaginary result bits at bit locations n and n-1.

$m = 0,1,2,3...,6$ and $n = 23,22,21,....16$
[Note, the m and n indexes are sequentially indexed simultaneously (e.g. when m=1, n=22)].

Any of the bits, RE_0-RE_6 or IM_0-IM_6, in the scale code can be set high by any one of the 4080 output data words. But, once a bit is set high, it must remain high for the remainder of the 4080 transform. All of the bits in the scale code are reset to 0 just prior to outputting the first word of the 4080 transform.

5.2.3.1 Scale Code Circuit. Hardware generation of each scale code bit is accomplished by the sequential finite state circuit shown in the top portion of Figure 5.12. Seven XORSCAL cells are required to produce the complete code. These cells are connected to the eight most

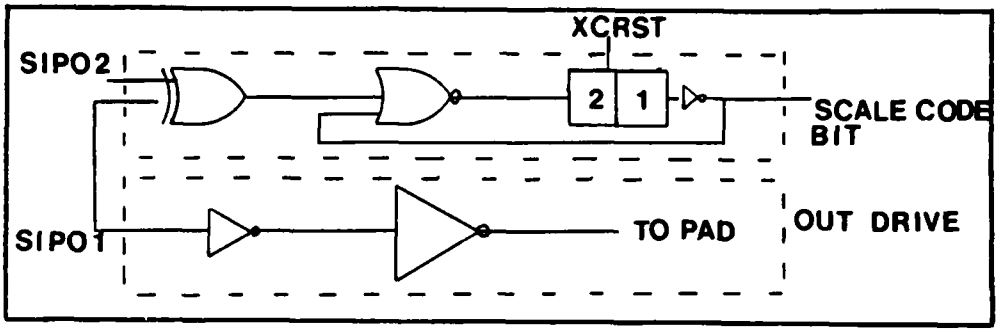


Figure 5.12 Logic Diagram of XORSCAL Cell

significant data word bits at the outputs of the SIPO array, see Figure 5.13. The functional operation of the circuit is depicted by the state diagram and truth table shown in Figure 5.14, and is summarized as follows:

a. The output of each XORSCAL cell is initialized to 0 just after the first set of 16 data words of the 4080-point transform are latched into the SIPO array. Initialization is achieved by the pulsed signal XCRST which sets the contents of the storage registers to 1, thereby making all of the scale bit outputs 0.

b. The next state of each output bit is a function of the NOR gate, whose inputs are the present output and the XOR gate output. As each data word is shifted out of the SIPO cells, the exclusive OR gate monitors adjacent bits to detect whether the two bits are different or the same. If the two bits are the same, a 0 is input into the NOR gate and the next output remains the same. However, if the two bits are different the next output will be a 1 and will remain at 1 until the circuit is reset at the beginning

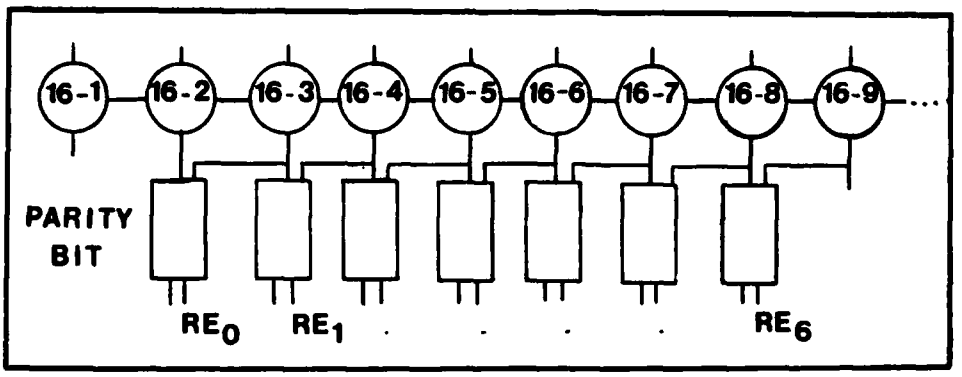
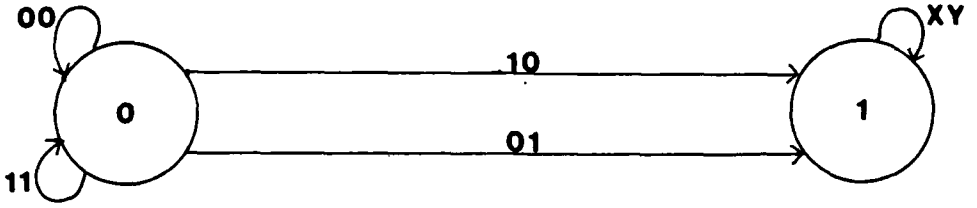


Figure 5.13 SIPO/XORSCAL Interface



(a)

XOR		NOR			
RB_n	RB_{n-1}	A	Present Output	Node B	Next Output
0	0	0	0	1	0
0	1	1	0	0	1
1	0	1	1	0	1
1	1	0	1	0	1

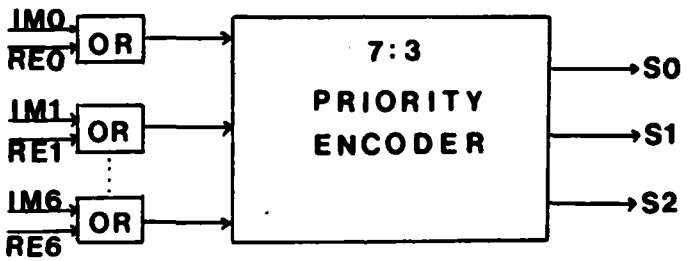
(b)

Figure 5.14 XORSCAL CKT (a) state diagram, (b) truth table.

of the next 4080-point transform. Therefore, an output of a 1 at the end of a 4080-point transform indicates that there is at least one data word where the most significant adjacent bit cannot be considered as a sign extension of the least significant adjacent bit.

5.2.3.2 Scale Factor Generation. The scale codes, IM_m and RE_m, produced by the XORSCAL cells, are transmitted to the scale factor generation circuit shown in Figure 5.15a. This circuit consists of seven OR gates and a priority encoder. The resulting scale factors are given in Figure 5.15b.

5.2.3.3 OUTDRIV Cell. This cell is simply two staged up inverters which are installed to insure the output pads are driven to the correct states in the allotted time. Note that the output drivers for data bits 15 through 22 are included as part of the XORSCAL cells.



(a)

X0	X1	X2	X3	X4	X5	X6	S0	S1	S2	Scale Factor
1	*	*	*	*	*	*	0	0	0	0
0	1	*	*	*	*	*	0	0	1	1
0	0	1	*	*	*	*	0	1	0	2
0	0	0	1	*	*	*	0	1	1	3
0	0	0	0	1	*	*	1	0	0	4
0	0	0	0	0	1	*	1	0	1	5
0	0	0	0	0	0	1	1	1	0	6
0	0	0	0	0	0	0	1	1	1	7

(b)

Figure 5.15 Scale Factor Generation (a) CKT, (b) Code Translation.

5.3 Macrocell Size and Density

Table 5.2 summarizes the macrocell sizes and density.

Cell Name	Arrayed Size (λ)	# Devices /cm^2
PISO	113 x 89	180,000
PARZF	124 x 89	129,00
SIPO	113 x 89	180,00
PARRND	197 X 103.5	129,000
XORSCALE	101 x 110	93,500

Table 5.2 Dimension/Density of I/O Macrocells
(3 micron technology)

VI. Conclusions and Recommendations

6.0 Conclusions

This thesis effort produced the design and implementation of 16 CMOS macrocells which are integrated into an efficient serial pipeline architecture to perform the Winograd Fourier Transform Algorithm at clocking rates in excess of 70 MHZ. At this rate, the architectures for the 15, 16, and 17-point WFTA can solve back to back WFT problems every 42.9 μs, 45.7 μs, and 48.6 μs, respectively. Efficient utilization of silicon area enables each of these architectures, with its associated control and optimized address circuitry, to be implemented on a single chip using a 1 1/4 micron process.

Power consumption was kept to a minimum by; (1) employing the CMOS technology, (2) making maximum use of dynamic master slave flip flops throughout the pipeline, and (3) designing an efficient adder/subtracter circuit whose devices (24 total) share drain and source areas.

Testability was achieved by building in an efficient means for test vector controllability and observability. The methodology employed is similar to the Level Sensitive Scan Design scheme. This scheme takes advantage of the serial pipeline architecture so the cost of testability in silicon area is less than 5 percent.

Fault tolerance of the overall 4080-point DFT VLSI system is enhanced by accomplishing parity checking on incoming data points and parity bit generation for transformed (output) data points.

The combined results of the four thesis efforts in the WFT are demonstrate that a VLSI system capable of computing over eight thousand (8000) 4080-point DFTs per second is a feasable goal.

6.1 Recommendations

Further research, design, and testing activities are required to bring the AFIT VLSI 4080-point DFT system to fruition. It is suggested that the next areas of emphasis be placed on test vector generation and clock propagation. The absence of research in these areas could prove to be significant barriers in testing and design validation.

Regarding the WFTA arithmetic circuitry: (1) additional SPICE simulations should be performed to further optimize the rise and fall times in the ADDSUB and PARRND cells, and, (2) the ADDUSB cell layout should be examined to identify alternate layouts that may reduce the required amount of silicon area.

Finally, a 3 micron chip should be developed to specifically test the PARZER, PARERR, PARRND, and XORSCAL cells.

Bibliography

1. Blahut, Richard E. Fast Algorithms for Digital Signal Processing. Reading: Addison-Wesley Publishing Company, 1985.

2. Chandramouli, R. "Designing VLSI Chips for Testability," Electronics Tests (USA), 5: 51-60 (November 1982).

3. Collins, James M. Simulation and Modeling of a VLSI Winograd Fourier Transform Processor. MS Thesis, AFIT/GE/ENG/85D-9. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1985.

4. Glasser, Lance A. and Daniel W. Dobberpuhl. The Design and Analysis of VLSI Circuits. Reading: Addison-Wesley Publishing Company, 1985.

5. Lewis, E.T. "CMOS Custom Circuit Design," Raytheon International Report, (December 1984).

6. Linderman, Richard W. and others. "CUSP: A 2-Micron CMOS Digital Signal Processor," IEEE Journal of Solid-State Circuits, SC-20, No. 3: 761-769 (June 1985).

7. Lyon, R.F. "Two's Complement Pipeline Multipliers," IEEE Transactions on Communications, 418 (April 1976).

8. Mead, Carver and Lynn Conway. Introduction to VLSI Systems. Reading: Addison-Wesley Publishing Company, 1980.

9. Oppenheim, Alan V. Digital Signal Processing. Englewood Cliffs: Prentice-Hall Inc., 1975.

10. Rabiner, Lawrence R. and Bernard Gold. Theory and Application of Digital Signal Processing. Englewood Cliffs: Prentice-Hall Inc., 1975.

11. Rossbach, Paul C. Control Circuitry for High Speed VLSI Winograd Fourier Transform Processors. MS Thesis, AFIT/GE/ENG/85D-39. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1985.

12. Taylor, Kent. Architecture and Numerical Accuracy of High Speed DFT Processing Systems. MS Thesis, AFIT/GE/ENG/85D-47. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1985.

13. Thompson, C. D. A Complexity Theory for VLSI. PhD dissertation. Carnegie-Mellon University, August 1980.

14. Weste, Neil H. E. and Kamran Eshraghian. Principles of CMOS VLSI Design. Reading: Addison-Wesley Publishing Company, 1985.

15. Cavanaugh, Joseph J.F. Digital Computer Arithmetic Design and Implementation. New York: McGraw-Hill Book Company, 1984.

VITA

Captain Paul W. Coutee was born on 31 March 1948 in Shreveport, Louisiana. He graduated from the Fenn College of Engineering, Cleveland State University, in June 1977 with a degree of Bachelor of Science in Electrical Engineering. He received a commission in the USAF through the OTS program, September 1977. He then served as an electronics systems engineer in the Air Training Command, Randolph AFB, Texas, until entering the School of Engineering, Air Force Institute of Technology, in June 1984.

Permanent address: 428 Redbud Lane
WPAFB, Ohio 45433

SECURITY CLASSIFICATION OF THIS PAGE

AD-A163943

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/85D-11	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Office of Scientific Research 8b. OFFICE SYMBOL (If applicable) AFOSR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) AFOSR Bolling AFB, Washington D.C. 20332	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Paul W. Coutee, Captain, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1985 December 15. PAGE COUNT 135
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	
09	02		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: ARITHMETIC CIRCUITRY FOR HIGH SPEED VLSI WINOGRAD FOURIER TRANSFORM PROCESSOR

Advisor: Richard W. Linderman, Capt, USAF
Assistant Professor of Electrical Engineering

Approved for public release: IAW AFR 190-1.
LYNN E. WOLAVER 15 JUN 86
Dean for Research and Professional Development
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB OH 45433

Fm 18. Subject Terms: Computer Architecture, and Digital Signal Processing, CMOS Arithmetic Circuitry, Winograd Fourier Transform

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION	
22a. NAME OF RESPONSIBLE INDIVIDUAL LINDERMAN	22b. TELEPHONE NUMBER (Include Area Code) 312-255-6913	22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

This investigation defines, designs, and implements the arithmetic circuitry for a VLSI system capable of computing over eight thousand (8000) 4080-point Discrete Fourier Transform (DFT) problems per second with high numerical accuracy (over 100 db). An overview of the architecture illustrates how it is segregated into three smaller DFT problems, the 15, 16, and 17-point DFTs, using the Good-Thomas Prime Factor Algorithm (PFA). The smaller DFT problems are solved, using the Winograd Fourier Transform Algorithm (WFTA), on separate VLSI chips.

The chips were designed for a 1.2 m CMOS process to operate at 70 MHZ. A detailed analysis of the serial pipeline architecture, used in the arithmetic sections to compute the WFTs, highlights the design of a constant coefficient serial pipeline multiplier. Each coefficient is encoded into its dual-bit (i.e., 0, ± 1, or ± 2 times 2^{2n}) equivalent and implemented using 6 multiplicand macrocells. The pipeline's pre- and post-addition stages feature a 24 device adder-subtractor cell designed to reduce the cell's load capacitance. The input and output circuitry perform: I/O data buffering, parity checking and generation, rounding, scale factor generation, and data word bit length extensions to prevent overflow and maintain numerical accuracy. Emphasis is placed on achieving a high measure of Fault Tolerance, Testability, and Low Power Consumption.

END

FILMED

3-86

DTIC